



Creating A Single Global Electronic Market

1
2
3 Collaboration-Protocol Profile and Agreement
4 Specification
5 Version 2.0

6
7 OASIS ebXML Collaboration Protocol Profile
8 and Agreement Technical Committee

9
10 September 23, 2002

11
12
13 **1 Status of this Document**

14
15 This document specifies an ebXML SPECIFICATION for the eBusiness community.

16
17 Distribution of this document is unlimited.

18
19 The document formatting is based on the Internet Society's Standard RFC format.

20
21 ***This version:***

22
23 http://www.oasis-open.org/committees/ebxml-cppa/documents/ebCPP-2_0.pdf

24
25 ***Errata for this version:***

26
27 http://www.oasis-open.org/committees/ebxml-cppa/documents/ebCPP-2_0-Errata.shtml

28
29 ***Previous version:***

30
31 <http://www.ebxml.org/specs/ebCCP.pdf>

33 **2 Technical Committee Members**

34	Selim Aissi	Intel
35	Arvola Chan	TIBCO
36	James Bryce Clark	Individual Member
37	David Fischer	Drummond Group
38	Tony Fletcher	Individual Member
39	Brian Hayes	Collaborative Domain
40	Neelakantan Kartha	Sterling Commerce
41	Kevin Liu	SAP
42	Pallavi Malu	Intel
43	Dale Moberg	Cyclone Commerce
44	Himagiri Mukkamala	Sybase
45	Peter Ogden	Cyclone Commerce
46	Marty Sachs	IBM
47	Yukinori Saito	Individual Member
48	David Smiley	Mercator
49	Tony Weida	Individual Member
50	Pete Wenzel	SeeBeyond
51	Jean Zheng	Vitria

52 **3 ebXML Participants**

53 The authors wish to recognize the following for their significant participation in developing the
54 Collaboration Protocol Profile and Agreement Specification, Version 1.0.

55

56 David Burdett, CommerceOne

57 Tim Chiou, United World Chinese Commercial Bank

58 Chris Ferris, Sun

59 Scott Hinkelman, IBM

60 Maryann Hondo, IBM

61 Sam Hunting, ECOM XML

62 John Ibbotson, IBM

63 Kenji Itoh, JASTPRO

64 Ravi Kacker, eXcelon Corp.

65 Thomas Limanek, iPlanet

66 Daniel Ling, VCHEQ

67 Henry Lowe, OMG

68 Dale Moberg, Cyclone Commerce

69 Duane Nickull, XML Global Technologies

70 Stefano Pogliani, Sun

71 Rebecca Reed, Mercator

72 Karsten Riemer, Sun

73 Marty Sachs, IBM

74 Yukinori Saito, ECOM

75 Tony Weida, Edifecs

76 **4 Table of Contents**

77	1	Status of this Document.....	1
78	2	Technical Committee Members.....	2
79	3	ebXML Participants.....	3
80	4	Table of Contents.....	4
81	5	Introduction.....	7
82	5.1	Summary of Contents of Document.....	7
83	5.2	Document Conventions.....	7
84	5.3	Versioning of the Specification and Schema.....	8
85	5.4	Definitions.....	9
86	5.5	Audience.....	9
87	5.6	Assumptions.....	9
88	5.7	Related Documents.....	9
89	6	Design Objectives.....	10
90	7	System Overview.....	11
91	7.1	What This Specification Does.....	11
92	7.2	Forming a CPA from Two CPPs.....	13
93	7.3	Forming a CPA from a CPA Template.....	15
94	7.4	How the CPA Works.....	15
95	7.5	Where the CPA May Be Implemented.....	16
96	7.6	Definition and Scope.....	17
97	8	CPP Definition.....	18
98	8.1	Globally-Unique Identifier of CPP Instance Document.....	19
99	8.2	CPP Structure.....	19
100	8.3	CollaborationProtocolProfile element.....	19
101	8.4	PartyInfo Element.....	20
102	8.4.1	PartyId element.....	22
103	8.4.2	PartyRef element.....	23
104	8.4.3	CollaborationRole element.....	24
105	8.4.4	ProcessSpecification element.....	26
106	8.4.5	Role element.....	30
107	8.4.6	ApplicationCertificateRef element.....	31
108	8.4.7	ApplicationSecurityDetailsRef element.....	31
109	8.4.8	ServiceBinding element.....	32
110	8.4.9	Service element.....	32
111	8.4.10	CanSend element.....	32
112	8.4.11	CanReceive element.....	33
113	8.4.12	ThisPartyActionBinding element.....	33
114	8.4.13	OtherPartyActionBinding.....	35
115	8.4.14	BusinessTransactionCharacteristics element.....	35
116	8.4.15	ChannelId element.....	38
117	8.4.16	ActionContext element.....	39
118	8.4.17	CollaborationActivity element.....	40
119	8.4.18	Certificate element.....	40
120	8.4.19	SecurityDetails element.....	41
121	8.4.20	TrustAnchors element.....	42
122	8.4.21	SecurityPolicy element.....	42
123	8.4.22	DeliveryChannel element.....	42
124	8.4.23	MessagingCharacteristics element.....	44
125	8.4.24	Transport element.....	48
126	8.4.25	TransportSender element.....	49
127	8.4.26	TransportProtocol element.....	49

128	8.4.27 AccessAuthentication element.....	49
129	8.4.28 TransportClientSecurity element.....	50
130	8.4.29 TransportSecurityProtocol element.....	50
131	8.4.30 ClientCertificateRef element.....	50
132	8.4.31 ServerSecurityDetailsRef element.....	51
133	8.4.32 Encryption Algorithm.....	51
134	8.4.33 TransportReceiver element.....	52
135	8.4.34 Endpoint element.....	52
136	8.4.35 TransportServerSecurity element.....	52
137	8.4.36 ServerCertificateRef element.....	53
138	8.4.37 ClientSecurityDetailsRef element.....	53
139	8.4.38 Transport protocols.....	53
140	8.4.39 DocExchange Element.....	56
141	8.4.40 ebXMLSenderBinding element.....	57
142	NOTE: A CPA could be valid even when omitting all children under <i>ebXMLSenderBinding</i>	57
143	8.4.41 ReliableMessaging element.....	57
144	8.4.42 PersistDuration element.....	58
145	8.4.43 SenderNonRepudiation element.....	59
146	8.4.44 NonRepudiationProtocol element.....	59
147	8.4.45 HashFunction element.....	59
148	8.4.46 SignatureAlgorithm element.....	59
149	8.4.47 SigningCertificateRef element.....	60
150	8.4.48 SenderDigitalEnvelope element.....	60
151	8.4.49 DigitalEnvelopeProtocol element.....	61
152	8.4.50 EncryptionAlgorithm element.....	61
153	8.4.51 EncryptionSecurityDetailsRef element.....	62
154	8.4.52 NamespaceSupported element.....	62
155	8.4.53 ebXMLReceiverBinding element.....	62
156	NOTE: A CPA could be valid even when omitting all children under <i>ebXMLReceiverBinding</i>	63
157	8.4.54 ReceiverNonRepudiation element.....	63
158	8.4.55 SigningSecurityDetailsRef element.....	63
159	8.4.56 ReceiverDigitalEnvelope element.....	63
160	8.4.57 EncryptionCertificateRef element.....	64
161	8.4.58 OverrideMshActionBinding element.....	64
162	8.5 SimplePart element.....	64
163	8.6 Packaging element.....	65
164	8.6.1 ProcessingCapabilities element.....	66
165	8.6.2 CompositeList element.....	66
166	8.7 Signature element.....	68
167	8.8 Comment element.....	69
168	9 CPA Definition.....	70
169	9.1 CPA Structure.....	70
170	9.2 CollaborationProtocolAgreement element.....	71
171	9.3 Status Element.....	71
172	9.4 CPA Lifetime.....	72
173	9.4.1 Start element.....	72
174	9.4.2 End element.....	72
175	9.5 ConversationConstraints Element.....	73
176	9.5.1 invocationLimit attribute.....	73
177	9.5.2 concurrentConversations attribute.....	74
178	9.6 PartyInfo Element.....	74
179	9.6.1 ProcessSpecification element.....	74
180	9.7 SimplePart element.....	75
181	9.8 Packaging element.....	75

182	9.9 Signature element	75
183	9.9.1 Persistent Digital Signature	76
184	9.10 Comment element.....	77
185	9.11 Composing a CPA from Two CPPs.....	77
186	9.11.1 ID Attribute Duplication.....	78
187	9.12 Modifying Parameters of the Process-Specification Document Based on Information in the CPA	78
188	10 References	80
189	11 Conformance	83
190	12 Disclaimer.....	84
191	13 Contact Information.....	85
192	Notices.....	87
193	Appendix A Example of CPP Document (Non-Normative).....	88
194	Appendix B Example of CPA Document (Non-Normative)	102
195	Appendix C Business Process Specification Corresponding to Complete CPP and CPA Definition (Non-Normative)	
196	115
197	Appendix D W3C XML Schema Document Corresponding to Complete CPP and CPA Definition (Normative)...	117
198	Appendix E CPA Composition (Non-Normative).....	126
199	E.1 Suggestions for Design of Computational Procedures	126
200	E.2 CPA Formation Component Tasks.....	128
201	E.3 CPA Formation from <i>CPPs</i> : Context of Tasks	128
202	E.4 Business Collaboration Process Matching Tasks	129
203	E.5 Implementation Matching Tasks	130
204	E.6 CPA Formation: Technical Details	148
205	Appendix F Correspondence Between CPA and ebXML Messaging Parameters (Normative).....	150
206	Appendix G Glossary of Terms	153

207 **5 Introduction**

208

209 **5.1 Summary of Contents of Document**

210 As defined in the ebXML Business Process Specification Schema[ebBPSS], a *Business Partner*
211 is an entity that engages in *Business Transactions* with another *Business Partner(s)*. The
212 *Message-exchange* capabilities of a *Party* MAY be described by a *Collaboration-Protocol*
213 *Profile (CPP)*. The *Message-exchange* agreement between two *Parties* MAY be described by a
214 *Collaboration-Protocol Agreement (CPA)*. A *CPA* MAY be created by computing the
215 intersection of the two *Partners' CPPs*. Included in the *CPP* and *CPA* are details of transport,
216 messaging, security constraints, and bindings to a *Business-Process-Specification* (or, for short,
217 *Process-Specification*) document that contains the definition of the interactions between the two
218 *Parties* while engaging in a specified electronic *Business Collaboration*.

219
220 This specification contains the detailed definitions of the *Collaboration-Protocol Profile (CPP)*
221 and the *Collaboration-Protocol Agreement (CPA)*.

222

223 This specification is a component of the suite of ebXML specifications.

224

225 The rest of this specification is organized as follows:

- 226 • Section 6 defines the objectives of this specification.
- 227 • Section 7 provides a system overview.
- 228 • Section 8 contains the definition of the *CPP*, identifying the structure and all necessary
229 fields.
- 230 • Section 9 contains the definition of the *CPA*.
- 231 • Section 10 lists all other documents referenced in this specification.
- 232 • Section 11 provides a conformance statement.
- 233 • Section 12 contains a disclaimer.
- 234 • Section 13 lists contact information for the contributing authors and the coordinating
235 editor for this version of the specification.
- 236 • The appendices include examples of *CPP* and *CPA* documents (non-normative), an
237 example XML *Business Process Specification* (non-normative), an XML Schema document
238 (normative), a description of how to compose a *CPA* from two *CPPs* (non-normative), a
239 summary of corresponding ebXML Messaging Service and *CPA* parameters (normative), and
240 a Glossary of Terms.

241 •

242 **5.2 Document Conventions**

243 Terms in *Italics* are defined in Appendix G (Glossary of Terms). Terms listed in ***Bold Italics***
244 represent the element and/or attribute content of the XML *CPP*, *CPA*, or related definitions.

245

246 In this specification, indented paragraphs beginning with "NOTE:" provide non-normative
247 explanations or suggestions that are not mandated by the specification.

248
249 References to external documents are represented with BLOCK text enclosed in brackets, e.g.
250 [RFC2396]. The references are listed in Section 10, "References".
251
252 The keywords MUST, MUST NOT, REQUIRED, SHALL, SHALL NOT, SHOULD, SHOULD
253 NOT, RECOMMENDED, MAY, and OPTIONAL, when they appear in this document, are to be
254 interpreted as described in [RFC 2119].

255
256 NOTE: Vendors SHOULD carefully consider support of elements with cardinalities (0 or
257 1) or (0 or more). Support of such an element means that the element is processed
258 appropriately for its defined function and not just recognized and ignored. A given *Party*
259 might use these elements in some *CPPs* or *CPAs* and not in others. Some of these elements
260 define parameters or operating modes and SHOULD be implemented by all vendors. It
261 might be appropriate to implement elective elements that represent major run-time
262 functions, such as various alternative communication protocols or security functions, by
263 means of plug-ins so that a given *Party* MAY acquire only the needed functions rather than
264 having to install all of them.

265
266 By convention, values of [XML] attributes are generally enclosed in quotation marks, however
267 those quotation marks are not part of the values themselves.
268

269 5.3 Versioning of the Specification and Schema

270 Whenever this specification is modified, it SHALL be given a new version number.

271
272 It is anticipated that during the review period, errors and inconsistencies in the specification and
273 in the schema may be detected and have to be corrected. All known errors in the specification as
274 well as necessary changes to the schema will be summarized in an errata page found at

275
276 http://www.oasis-open.org/committees/ebxml-cppa/documents/ebCPP-2_0-Errata.shtml

277
278 The specification, when initially approved by the OASIS ebXML Collaboration Protocol Profile
279 and Agreement Technical Committee for public review, SHALL carry a version number of
280 "2_0". At that time, the schema SHALL have a version number of "2_0b" and the suffix letter
281 after "2_0" will be advanced as necessary when bug fixes to the schema have to be introduced.
282 Such versions of the schema SHALL be found under the directory

283
284 <http://www.oasis-open.org/committees/ebxml-cppa/schema/>

285
286 In addition, the latest version of the schema SHALL always be found at

287
288 http://www.oasis-open.org/committees/ebxml-cppa/schema/cpp-cpa-2_0.xsd

289
290 since the latter is the namespace URI used for this specification and the corresponding schema is
291 supposed to be directly resolvable from the namespace URI.

292
293 The value of the version attribute of the Schema element in a given version of the schema
294 SHALL be equal to the version of the schema.
295

296 **5.4 Definitions**

297 Technical terms in this specification are defined in Appendix G.
298

299 **5.5 Audience**

300 One target audience for this specification is implementers of ebXML services and other
301 designers and developers of middleware and application software that is to be used for
302 conducting electronic *Business*. Another target audience is the people in each enterprise who are
303 responsible for creating *CPPs* and *CPAs*.
304

305 **5.6 Assumptions**

306 It is expected that the reader has an understanding of XML and is familiar with the concepts of
307 electronic *Business* (eBusiness).
308

309 **5.7 Related Documents**

310 Related documents include ebXML Specifications on the following topics:
311

- 312 • ebXML *Message* Service Specification[ebMS]
- 313 • ebXML Business Process Specification Schema[ebBPSS]
- 314 • ebXML Core Component Overview[ccOVER]
- 315 • ebXML Registry Services Specification[ebRS]

316
317 See Section 10 for the complete list of references.
318

319 **6 Design Objectives**

320 The objective of this specification is to ensure interoperability between two *Parties* even though
321 they MAY procure application software and run-time support software from different vendors.
322 The *CPP* defines a *Party's Message-exchange* capabilities and the *Business Collaborations* that
323 it supports. The *CPA* defines the way two *Parties* will interact in performing the chosen *Business*
324 *Collaborations*. Both *Parties* SHALL use identical copies of the *CPA* to configure their run-
325 time systems. This assures that they are compatibly configured to exchange *Messages* whether or
326 not they have obtained their run-time systems from the same vendor. The configuration process
327 MAY be automated by means of a suitable tool that reads the *CPA* and performs the
328 configuration process.

329
330 In addition to supporting direct interaction between two *Parties*, this specification MAY also be
331 used to support interaction between two *Parties* through an intermediary such as a portal or
332 broker.

333
334 It is an objective of this specification that a *CPA* SHALL be capable of being composed by
335 intersecting the respective *CPPs* of the *Parties* involved. The resulting *CPA* SHALL contain
336 only those elements that are in common, or compatible, between the two *Parties*. Variable
337 quantities, such as number of retries of errors, are then negotiated between the two *Parties*. The
338 design of the *CPP* and *CPA* schemata facilitates this composition/negotiation process. However,
339 the composition and negotiation processes themselves are outside the scope of this specification.
340 Appendix E contains a non-normative discussion of this subject.

341
342 It is a further objective of this specification to facilitate migration of both traditional EDI-based
343 applications and other legacy applications to platforms based on the ebXML specifications. In
344 particular, the *CPP* and *CPA* are components of the migration of applications based on the X12
345 838 Trading-Partner Profile[X12] to more automated means of setting up *Business* relationships
346 and doing *Business* under them.

347 **7 System Overview**

348 **7.1 What This Specification Does**

349 The exchange of information between two *Parties* requires each *Party* to know the other *Party's*
350 supported *Business Collaborations*, the other *Party's* role in the *Business Collaboration*, and the
351 technology details about how the other *Party* sends and receives *Messages*. In some cases, it is
352 necessary for the two *Parties* to reach agreement on some of the details.

353
354 The way each *Party* can exchange information, in the context of a *Business Collaboration*, can
355 be described by a *Collaboration-Protocol Profile (CPP)*. The agreement between the *Parties* can
356 be expressed as a *Collaboration-Protocol Agreement (CPA)*.

357
358 A *Party* MAY describe itself in a single *CPP*. A *Party* MAY create multiple *CPPs* that describe,
359 for example, different *Business Collaborations* that it supports, its operations in different regions
360 of the world, or different parts of its organization.

361
362 To enable *Parties* wishing to do *Business* to find other *Parties* that are suitable *Business*
363 *Partners*, *CPPs* MAY be stored in a repository such as is provided by the ebXML
364 Registry[ebRS]. Using a discovery process provided as part of the specifications of a repository,
365 a *Party* MAY then use the facilities of the repository to find *Business Partners*.

366
367 The document that defines the interactions between two *Parties* is a *Process-Specification*
368 document that MAY conform to the ebXML Business Process Specification Schema[ebBPSS].
369 The *CPP* and *CPA* include references to this *Process-Specification* document. The *Process-*
370 *Specification* document MAY be stored in a repository such as the ebXML Registry. See NOTE
371 about alternative *Business-Collaboration* descriptions in Section 8.4.4.

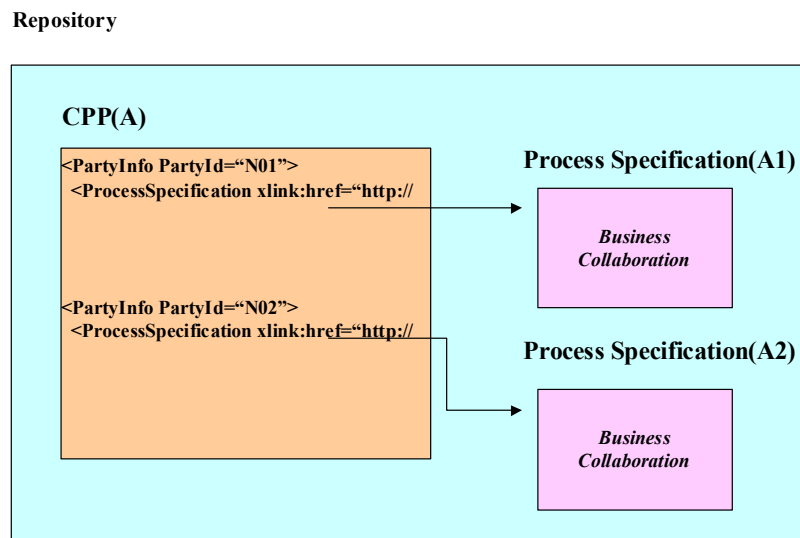
372
373 Figure 1 illustrates the relationships between a *CPP* and two *Process-Specification* documents,
374 A1 and A2, in an ebXML Registry. On the left is a *CPP*, A, which includes information about
375 two parts of an enterprise that are represented as different *Parties*. On the right are shown two
376 *Process-Specification* documents. Each of the ***PartyInfo*** elements in the *CPP* contains a
377 reference to one of the *Process-Specification* documents. This identifies the *Business*
378 *Collaborations* that the *Party* can perform.

379
380 This specification defines the markup language vocabulary for creating electronic *CPPs* and
381 *CPAs*. *CPPs* and *CPAs* are [XML] documents. In the appendices of this specification are two
382 sample *CPPs*, a sample *CPA* formed from the *CPPs*, a sample *Process-Specification* referenced
383 by the *CPPs* and the *CPA*, and the XML Schema governing the structures of *CPPs* and *CPAs*.

384
385 The *CPP* describes the capabilities of an individual *Party*. A *CPA* describes the capabilities that
386 two *Parties* have agreed to use to perform particular *Business Collaborations*. These *CPAs*
387 define the "information technology terms and conditions" that enable *Business* documents to be
388 electronically interchanged between *Parties*. The information content of a *CPA* is similar to the
389 information-technology specifications sometimes included in Electronic Data Interchange (EDI)

390 *Trading Partner Agreements (TPAs)*. However, these *CPAs* are not paper documents. Rather,
 391 they are electronic documents that can be processed by computers at the *Parties'* sites in order to
 392 set up and then execute the desired *Business* information exchanges. The "legal" terms and
 393 conditions of a *Business* agreement are outside the scope of this specification and therefore are
 394 not included in the *CPP* and *CPA*.

Figure 1: Structure of CPP & Business Process Specification in an ebXML Registry



395
 396 An enterprise MAY choose to represent itself as multiple *Parties*. For example, it might
 397 represent a central office supply procurement organization and a manufacturing supplies
 398 procurement organization as separate *Parties*. The enterprise MAY then construct a *CPP* that
 399 includes all of its units that are represented as separate *Parties*. In the *CPP*, each of those units
 400 would be represented by a separate *PartyInfo* element.

401
 402 The CPPA specification is concerned with software that conducts business on behalf of *Parties*
 403 by exchanging *Messages*[ebMS]. In particular, it is concerned with *Client* and *Server* software
 404 programs that engage in *Business Transactions*[ebBPSS] by sending and receiving *Messages*.
 405 Those *Messages* convey *Business Documents* and/or business signals[ebBPSS] in their payload.
 406 Under the terms of a *CPA*:

- 407
 408
- A *Client* initiates a connection with a *Server*.
 - A *Requester* initiates a *Business Transaction* with a *Responder*.
 - A *Sender* sends a *Message* to a *Receiver*.
- 409
 410

411
 412 Thus, *Client* and *Server* are software counterparts, *Requester* and *Responder* are business
 413 counterparts, and *Sender* and *Receiver* are messaging counterparts. There is no fixed

414 relationship between counterparts of different types. For example, consider a purchasing
415 collaboration. *Client* software representing the buying party might connect to *Server* software
416 representing the selling party, and then make a purchase request by sending a *Message*
417 containing a purchase order over that connection. If the CPA specifies a synchronous business
418 response, the *Server* might then respond by sending a *Message* containing an acceptance notice
419 back to the *Client* over the same connection. Alternatively, if the CPA specifies an
420 asynchronous business response, *Client* software representing the selling party might later
421 respond by connecting to *Server* software representing the buying party and then sending a
422 *Message* containing an acceptance notice.

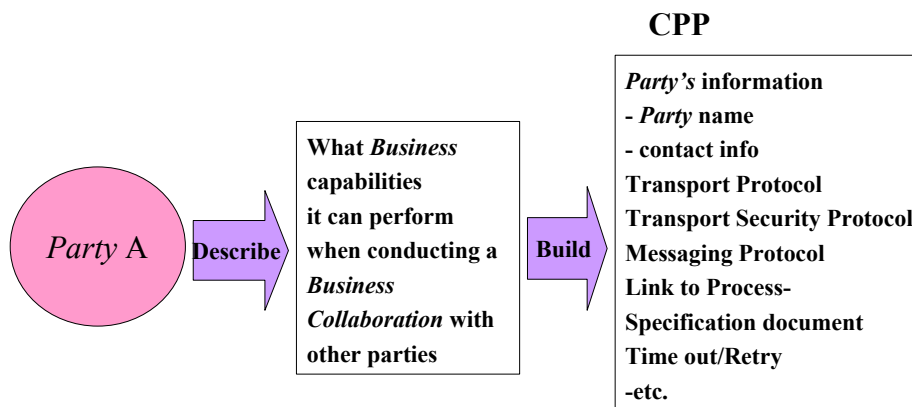
423
424 In general, the *Parties* to a *CPA* can have both client and server characteristics. A client requests
425 services and a server provides services to the *Party* requesting services. In some applications,
426 one *Party* only requests services and one *Party* only provides services. These applications have
427 some resemblance to traditional client-server applications. In other applications, each *Party*
428 MAY request services of the other. In that case, the relationship between the two *Parties* can be
429 described as a peer-peer relationship rather than a client-server relationship.
430

431 **7.2 Forming a CPA from Two CPPs**

432 This section summarizes the process of discovering a *Party* to do *Business* with and forming a
433 *CPA* from the two *Parties'* *CPPs*. In general, this section is an overview of a possible procedure
434 and is not to be considered a normative specification. See Appendix E "CPA Composition (Non-
435 Normative)" for more information.

436
437 Figure 2 illustrates forming a *CPP*. *Party A* tabulates the information to be placed in a repository
438 for the discovery process, constructs a *CPP* that contains this information, and enters it into an
439 ebXML Registry or similar repository along with additional information about the *Party*. The
440 additional information might include a description of the *Businesses* that the *Party* engages in.
441 Once *Party A's* information is in the repository, other *Parties* can discover *Party A* by using the
442 repository's discovery services.

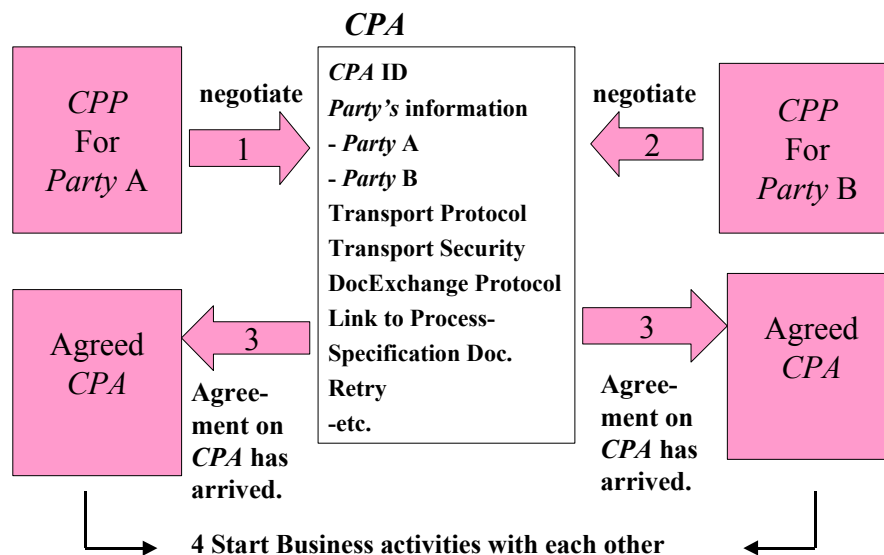
Figure 2: Overview of Collaboration-Protocol Profiles (CPP)



443

444 In Figure 3, *Party A* and *Party B* use their *CPPs* to jointly construct a single copy of a *CPA* by
 445 calculating the intersection of the information in their *CPPs*. The resulting *CPA* defines how the
 446 two *Parties* will behave in performing their *Business Collaboration*.

Figure 3: Overview of Collaboration-Protocol Agreements (CPA)

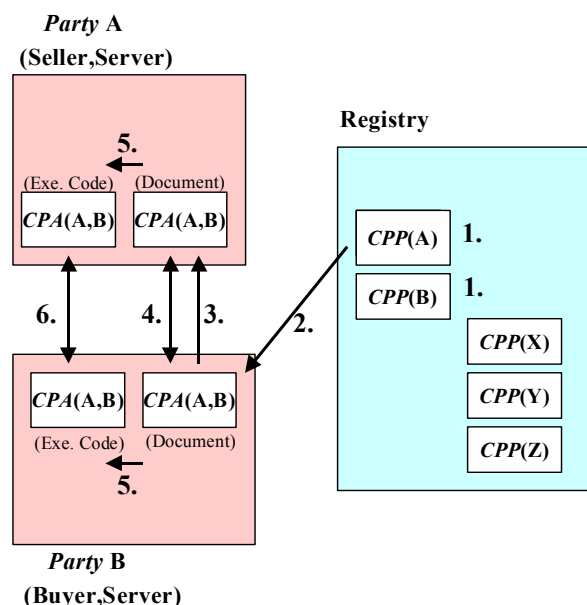


447

448 Figure 4 illustrates the entire process. The steps are listed at the left. The end of the process is
449 that the two *Parties* configure their systems from identical copies of the agreed *CPA* and they are

Figure 4: Overview of Working Architecture of CPP/CPA with ebXML Registry

1. Any *Party* may register its CPPs to an ebXML Registry.
2. *Party B* discovers trading partner A (Seller) by searching in the Registry and downloads *CPP(A)* to *Party B*'s server.
3. *Party B* creates *CPA(A,B)* and sends *CPA(A,B)* to *Party A*.
4. *Parties A* and *B* negotiate and store identical copies of the completed *CPA* as a document in both servers. This process is done manually or automatically.
5. *Parties A* and *B* configure their run-time systems with the information in the *CPA*.
6. *Parties A* and *B* do business under the new *CPA*.



450 then ready to do *Business*.

451

452 7.3 Forming a CPA from a CPA Template

453 Alternatively, a *CPA* template might be used to create a *CPA*. A *CPA* template represents one
454 party's "fill in the blanks" proposal to a prospective trading partner for implementing one or
455 more business processes. For example, such a template might contain placeholder values for
456 identifying aspects of the other party. To form a *CPA* from a *CPA* template, the placeholder
457 values would be replaced by the actual values for the other trading partner. Actual values might
458 be obtained from the other party's CPP, if available, or by data entry in an HTML form, among
459 other possibilities. The current version of this specification does not address how placeholder
460 values might be represented in a *CPA*. However, the process of filling out a *CPA* template
461 MUST result in a valid *CPA*. Further discussion of *CPA* templates is provided in Appendix E.
462

463 7.4 How the CPA Works

464 A *CPA* describes all the valid visible, and hence enforceable, interactions between the *Parties*
465 and the way these interactions are carried out. It is independent of the internal processes executed
466 at each *Party*. Each *Party* executes its own internal processes and interfaces them with the
467 *Business Collaboration* described by the *CPA* and *Process-Specification* document. The *CPA*
468 does not expose details of a *Party's* internal processes to the other *Party*. The intent of the *CPA* is

Collaboration-Protocol Profile and Agreement Specification

Page

469 to provide a high-level specification that can be easily comprehended by humans and yet is
470 precise enough for enforcement by computers.

471
472 The information in the *CPA* is used to configure the *Parties'* systems to enable exchange of
473 *Messages* in the course of performing the selected *Business Collaboration*. Typically, the
474 software that performs the *Message* exchanges and otherwise supports the interactions between
475 the *Parties* is middleware that can support any selected *Business Collaboration*. One component
476 of this middleware MAY be the ebXML *Message Service Handler*[ebMS]. In this specification,
477 the term "run-time system" or "run-time software" is used to denote such middleware.

478
479 The *CPA* and the *Process-Specification* document that it references define a conversation
480 between the two *Parties*. The conversation represents a single unit of *Business* as defined by the
481 **BinaryCollaboration** component of the *Process-Specification* document. The conversation
482 consists of one or more *Business Transactions*, each of which is a request *Message* from one
483 *Party* and zero or one response *Message* from the other *Party*. The *Process-Specification*
484 document defines, among other things, the request and response *Messages* for each *Business*
485 *Transaction* and the order in which the *Business Transactions* are REQUIRED to occur. See
486 [ebBPSS] for a detailed explanation.

487
488 The *CPA* MAY actually reference more than one *Process-Specification* document. When a *CPA*
489 references more than one *Process-Specification* document, each *Process-Specification* document
490 defines a distinct type of conversation. Any one conversation involves only a single *Process-*
491 *Specification* document.

492
493 A new conversation is started each time a new unit of *Business* is started. The *Business*
494 *Collaboration* also determines when the conversation ends. From the viewpoint of a *CPA*
495 between *Party A* and *Party B*, the conversation starts at *Party A* when *Party A* sends the first
496 request *Message* to *Party B*. At *Party B*, the conversation starts when it receives the first request
497 of the unit of *Business* from *Party A*. A conversation ends when the *Parties* have completed the
498 unit of *Business*.

499
500 NOTE: The run-time system SHOULD provide an interface by which the *Business*
501 application can request initiation and ending of conversations.
502

503 **7.5 Where the CPA May Be Implemented**

504 Conceptually, a *Business-to-Business* (B2B) server at each *Party's* site implements the *CPA* and
505 *Process-Specification* document. The B2B server includes the run-time software, i.e. the
506 middleware that supports communication with the other *Party*, execution of the functions
507 specified in the *CPA*, interfacing to each *Party's* back-end processes, and logging the interactions
508 between the *Parties* for purposes such as audit and recovery. The middleware might support the
509 concept of a long-running conversation as the embodiment of a single unit of *Business* between
510 the *Parties*. To configure the two *Parties'* systems for *Business-to-Business* operations, the
511 information in the copy of the *CPA* and *Process-Specification* documents at each *Party's* site is
512 installed in the run-time system. The static information MAY be recorded in a local database and

513 other information in the *CPA* and *Process-Specification* document MAY be used in generating or
514 customizing the necessary code to support the *CPA*.

515
516 NOTE: It is possible to provide a graphical *CPP/CPA*-authoring tool that understands both
517 the semantics of the *CPP/CPA* and the XML syntax. Equally important, the definitions in
518 this specification make it feasible to automatically generate, at each *Party's* site, the code
519 needed to execute the *CPA*, enforce its rules, and interface with the *Party's* back-end
520 processes.

521

522 **7.6 Definition and Scope**

523 This specification defines and explains the contents of the *CPP* and *CPA* XML documents. Its
524 scope is limited to these definitions. It does not define how to compose a *CPA* from two *CPPs*
525 nor does it define anything related to run-time support for the *CPP* and *CPA*. It does include
526 some non-normative suggestions and recommendations regarding *CPA* composition from two
527 *CPPs* and run-time support where these notes serve to clarify the *CPP* and *CPA* definitions. See
528 Section 11 for a discussion of conformance to this specification.

529

530 NOTE: This specification is limited to defining the contents of the *CPP* and *CPA*, and it
531 is possible to be conformant with it merely by producing a *CPP* or *CPA* document that
532 conforms to the XML Schema document defined herein. It is, however, important to
533 understand that the value of this specification lies in its enabling a run-time system that
534 supports electronic commerce between two *Parties* under the guidance of the information
535 in the *CPA*.

536 **8 CPP Definition**

537 A *CPP* defines the capabilities of a *Party* to engage in electronic *Business* with other *Parties*.
538 These capabilities include both technology capabilities, such as supported communication and
539 messaging protocols, and *Business* capabilities in terms of what *Business Collaborations* it
540 supports.

541
542 This section defines and discusses the details in the *CPP* in terms of the individual XML
543 elements. The discussion is illustrated with some XML fragments. See Appendix D for the XML
544 Schema, and Appendix A for sample *CPP* documents.

545
546 The ***ProcessSpecification***, ***DeliveryChannel***, ***DocExchange***, and ***Transport*** elements of the
547 *CPP* describe the processing of a unit of *Business* (conversation). These elements form a layered
548 structure somewhat analogous to a layered communication model.

549
550 **Process-Specification layer** - The *Process-Specification* layer defines the heart of the *Business*
551 agreement between the *Parties*: the services (*Business Transactions*) which *Parties* to the *CPA*
552 can request of each other and transition rules that determine the order of requests. This layer is
553 defined by the separate *Process-Specification* document that is referenced by the *CPP* and *CPA*.
554

555 **Delivery Channels** - A delivery channel describes a *Party's* *Message*-receiving and *Message*-
556 sending characteristics. It consists of one document-exchange definition and one transport
557 definition. Several delivery channels MAY be defined in one *CPP*.
558

559 **Document-Exchange Layer** - The Document-exchange layer specifies processing of the
560 business documents by the Message-exchange function. Properties specified include encryption,
561 digital signature, and reliable-messaging characteristics. The options selected for the Document-
562 exchange layer are complementary to those selected for the transport layer. For example, if
563 Message security is desired and the selected transport protocol does not provide *Message*
564 encryption, then *Message* encryption MUST be specified in the Document-exchange layer. The
565 protocol for exchanging *Messages* between two *Parties* is defined by the ebXML Message
566 Service specification[ebMS] or other similar messaging services.
567

568 **Transport layer** - The transport layer identifies the transport protocol to be used in sending
569 messages through the network and defines the endpoint addresses, along with various other
570 properties of the transport protocol. Choices of properties in the transport layer are
571 complementary to those in the document-exchange layer (see "Document-Exchange Layer"
572 directly above.)
573

574 Note that the functional layers encompassed by the *CPP* are independent of the contents of the
575 payload of the *Business* documents.
576

577 8.1 Globally-Unique Identifier of CPP Instance Document

578 When a *CPP* is placed in an ebXML or other Registry, the Registry assigns it a globally unique
579 identifier (GUID) that is part of its metadata. That GUID MAY be used to distinguish among
580 *CPPs* belonging to the same *Party*.

581
582 NOTE: A Registry cannot insert the GUID into the *CPP*. In general, a Registry does not
583 alter the content of documents submitted to it. Furthermore, a *CPP* MAY be signed and
584 alteration of a signed *CPP* would invalidate the signature.
585

586 8.2 CPP Structure

587 Following is the overall structure of the *CPP*. Unless otherwise noted, *CPP* elements MUST be
588 in the order shown here. Subsequent sections describe each of the elements in greater detail.

```
589
590 <tp:CollaborationProtocolProfile
591   xmlns:tp="http://www.oasis-open.org/committees/ebxml-
592 cppa/schema/cpp-cpa-2_0.xsd"
593   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
594   xsi:schemaLocation="http://www.oasis-open.org/committees/ebxml-
595 cppa/schema/cpp-cpa-2_0.xsd http://www.oasis-open.org/committees/ebxml-
596 cppa/schema/cpp-cpa-2_0.xsd"
597   xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
598   xmlns:xlink="http://www.w3.org/1999/xlink"
599   tp:cppid="uri:companyA-cpp"
600   tp:version="2_0b">
601   <tp:PartyInfo> <!-- one or more -->
602     ...
603   </tp:PartyInfo>
604   <tp:SimplePart id="..."> <!-- one or more -->
605     ...
606   </tp:SimplePart>
607   <tp:Packaging id="..."> <!-- one or more -->
608     ...
609   </tp:Packaging>
610   <tp:Signature> <!-- zero or one -->
611     ...
612   </tp:Signature>
613   <tp:Comment>text</tp:Comment> <!-- zero or more -->
614 </tp:CollaborationProtocolProfile>
615
```

616 8.3 CollaborationProtocolProfile element

617 The *CollaborationProtocolProfile* element is the root element of the *CPP* XML document.

618 The REQUIRED XML [XML] Namespace[XMLNS] declarations for the basic document are as
619 follows:

- 620 • The *CPP/CPA* namespace: xmlns:tp="http://www.oasis-open.org/committees/ebxml-
621 cppa/schema/cpp-cpa-2_0.xsd",
- 622 • The XML Digital Signature namespace:
623 xmlns:ds="http://www.w3.org/2000/09/xmldsig#",
- 624 • and the XLink namespace: xmlns:xlink="http://www.w3.org/1999/xlink".
625

626 In addition, the *CollaborationProtocolProfile* element contains a REQUIRED *cppid* attribute
 627 that supplies a unique identifier for the document, plus a REQUIRED *version* attribute that
 628 indicates the version of the schema. Its purpose is to identify the version of the schema that the
 629 *CPP* conforms to. The value of the *version* attribute SHOULD be a string such as "2_0a",
 630 "2_0b", etc.

631
 632 NOTE: The method of assigning unique *cppid* values is left to the implementation.

633
 634 The *CollaborationProtocolProfile* element SHALL consist of the following child elements:

- 635 • One or more REQUIRED *PartyInfo* elements that identify the organization (or parts of
- 636 the organization) whose capabilities are described by the *CPP*,
- 637 • One or more REQUIRED *SimplePart* elements that describe the constituents used to
- 638 make up composite *Messages*,
- 639 • One or more REQUIRED *Packaging* elements that describe how the *Message Header*
- 640 and payload constituents are packaged for transmittal,
- 641 • Zero or one *Signature* element that contains the digital signature that signs the *CPP*
- 642 document,
- 643 • Zero or more *Comment* elements.
- 644 •

645 A *CPP* document MAY be digitally signed so as to provide for a means of ensuring that the
 646 document has not been altered (integrity) and to provide for a means of authenticating the author
 647 of the document. A digitally signed *CPP* SHALL be signed using technology that conforms to
 648 the joint W3C/IETF XML Digital Signature specification[XMLDSIG].
 649

650 8.4 PartyInfo Element

651 The *PartyInfo* element identifies the organization whose capabilities are described in this *CPP*
 652 and includes all the details about this *Party*. More than one *PartyInfo* element MAY be
 653 provided in a *CPP* if the organization chooses to represent itself as subdivisions with different
 654 characteristics. Each of the sub-elements of *PartyInfo* is discussed later. The overall structure of
 655 the *PartyInfo* element is as follows:

```

656
657 <tp:PartyInfo
658   tp:partyName="..."
659   tp:defaultMshChannelId="..."
660   tp:defaultMshPackageId="...">
661   <tp:PartyId tp:type="..."> <!-- one or more -->
662     ...
663   </tp:PartyId>
664   <tp:PartyRef xlink:href="..." />
665   <tp:CollaborationRole> <!-- one or more -->
666     ...
667   </tp:CollaborationRole>
668   <tp:Certificate> <!-- one or more -->
669     ...
670   </tp:Certificate>
671   <tp:SecurityDetails> <!-- one or more -->
672     ...
673   </tp:SecurityDetails>
674   <tp:DeliveryChannel> <!-- one or more -->
```

```

675     ...
676     </tp:DeliveryChannel>
677     <tp:Transport> <!-- one or more -->
678     ...
679     </tp:Transport>
680     <tp:DocExchange> <!-- one or more -->
681     ...
682     </tp:DocExchange>
683     <tp:OverrideMshActionBinding> <!-- zero or more -->
684     ...
685     </tp:OverrideMshActionBinding>
686 </tp:PartyInfo>
687

```

688 The **PartyInfo** element contains a REQUIRED *partyName* attribute that indicates the common,
689 human readable name of the organization. Unlike **PartyID**, *partyName* might not be unique;
690 however, the value of each *partyName* attribute SHALL be meaningful enough to directly
691 identify the organization or the subdivision of an organization described in the **PartyInfo**
692 element.

693
694 The following example illustrates two possible party names.

```

695     <tp:PartyInfo tp:partyName="Example, Inc."...</tp:PartyInfo>
696
697     <tp:PartyInfo tp:partyName="Example, Inc. US Western Division">
698     ...
699     </tp:PartyInfo>
700

```

701
702 The **PartyInfo** element also contains a REQUIRED *defaultMshChannelId* attribute and a
703 REQUIRED *defaultMshPackageId* attribute. The *defaultMshChannelId* attribute identifies the
704 default **DeliveryChannel** to be used for sending standalone *Message Service Handler*[ebMS]
705 level messages (i.e., Acknowledgment, Error, StatusRequest, StatusResponse, Ping, Pong) that
706 are to be delivered asynchronously. When synchronous reply mode is in use, *Message Service*
707 *Handler* level messages are by default returned synchronously. The default can be overridden
708 through the use of **OverrideMshActionBinding** elements. The *defaultMshPackageId* attribute
709 identifies the default Packaging to be used for sending standalone *Message Service*
710 *Handler*[ebMS] level messages.

711
712 The **PartyInfo** element consists of the following child elements:

- 713 • One or more REQUIRED **PartyId** elements that provide logical identifiers for the
714 organization.
- 715 • One or more REQUIRED **PartyRef** elements that provide pointers to more information
716 about the *Party*.
- 717 • One or more REQUIRED **CollaborationRole** elements that identify the roles that this
718 *Party* can play in the context of a *Process Specification*.
- 719 • One or more REQUIRED **Certificate** elements that identify the certificates used by this
720 *Party* in security functions.
- 721 • One or more REQUIRED **SecurityDetails** elements that identify trust anchors and
722 specify security policy used by this *Party* in security functions.
- 723 • One or more REQUIRED **DeliveryChannel** elements that define the characteristics that
724 the *Party* can use to send and/or receive *Messages*. It includes both the transport protocol

- 725 (e.g. HTTP) and the messaging protocol (e.g. ebXML *Message Service*).
- 726 • One or more REQUIRED **Transport** elements that define the characteristics of the
 - 727 transport protocol(s) that the *Party* can support to send and/or receive *Messages*.
 - 728 • One or more REQUIRED **DocExchange** elements that define the *Message-exchange*
 - 729 characteristics, such as the signature and encryption protocols, that the *Party* can support.
 - 730 • Zero or more **OverrideMshActionBinding** elements that specify the *DeliveryChannel* to
 - 731 use for asynchronously delivered *Message Service Handler* level messages.

732 8.4.1 PartyId element

734 The REQUIRED **PartyId** element provides an identifier that SHALL be used to logically
 735 identify the *Party*. Additional **PartyId** elements MAY be present under the same **PartyInfo**
 736 element so as to provide for alternative logical identifiers for the *Party*. If the *Party* has
 737 preferences as to which logical identifier is used, the **PartyId** elements SHOULD be listed in
 738 order of preference starting with the most-preferred identifier.

739 In a *CPP* that contains multiple **PartyInfo** elements, different **PartyInfo** elements MAY contain
 741 **PartyId** elements that define different logical identifiers. This permits a large organization, for
 742 example, to have different identifiers for different purposes.

743 The value of the **PartyId** element is any string that provides a unique identifier. The identifier
 744 MAY be any identifier that is understood by both *Parties* to a *CPA*. Typically, the identifier
 745 would be listed in a well-known directory such as DUNS (Dun and Bradstreet) or in any naming
 746 system specified by [ISO6523].

747 The **PartyId** element has a single IMPLIED attribute: **type** that has an anyURI [XMLSCHEMA-
 748 2] value.

749 If the **type** attribute is present, then it provides a scope or namespace for the content of the
 750 **PartyId** element.

751 If the **type** attribute is not present, the content of the **PartyId** element MUST be a URI that
 752 conforms to [RFC2396]. It is RECOMMENDED that the value of the **type** attribute be a URN
 753 that defines a namespace for the value of the **PartyId** element. Typically, the URN would be
 754 registered in a well-known directory of organization identifiers.

755 The following example illustrates two URI references.

```
761 <tp:PartyId tp:type="urn:oasis:names:tc:ebxml-cppa:partyid-  
762 type:duns">123456789</tp:PartyId>  
763  
764 <tp:PartyId>urn:icann:example.com</tp:PartyId>
```

765 The first example is the *Party's* DUNS number. The value is the DUNS number of the
 766 organization.

767

770 The second example shows an arbitrary URN. This might be a URN that the *Party* has
771 registered with IANA, the Internet Assigned Numbers Authority (<http://www.iana.org>) to
772 identify itself directly.

773

774 The following document discusses naming agencies and how they are identified via URI values
775 of the *type* attribute:

776

777 http://www.oasis-open.org/committees/ebxml-cppa/documents/PartyID_Types.shtml

778

779 **8.4.2 PartyRef element**

780 The *PartyRef* element provides a link, in the form of a URI, to additional information about the
781 *Party*. Typically, this would be the URL from which the information can be obtained. The
782 information might be at the *Party's* web site or in a publicly accessible repository such as an
783 ebXML Registry, a UDDI repository (www.uddi.org), or a Lightweight Directory Access
784 Protocol[RFC2251] (LDAP) directory. Information available at that URI MAY include contact
785 information like names, addresses, and phone numbers, or context information like geographical
786 locales and industry segments, or perhaps more information about the *Business Collaborations*
787 that the *Party* supports. This information MAY be in the form of an ebXML Core
788 Component[ccOVER]. It is not within the scope of this specification to define the content or
789 format of the information at that URI.

790

791 The *PartyRef* element is an [XLINK] simple link. It has the following attributes:

- 792 • a FIXED *xlink:type* attribute,
- 793 • a REQUIRED *xlink:href* attribute,
- 794 • an IMPLIED *type* attribute,
- 795 • an IMPLIED *schemaLocation* attribute.

796

797 The contents of the document referenced by the *partyRef* element are subject to change at any
798 time. Therefore, it SHOULD NOT be cached for a long period of time. Rather, the value of the
799 *xlink:href* SHOULD be dereferenced only when the contents of this document are needed.

800

801 **8.4.2.1 xlink:type attribute**

802 The FIXED *xlink:type* attribute SHALL have a value of "simple". This identifies the element as
803 being an [XLINK] simple link.

804

805 **8.4.2.2 xlink:href attribute**

806 The REQUIRED *xlink:href* attribute SHALL have a value that is a URI that conforms to
807 [RFC2396] and identifies the location of the external information about the *Party*.

808

809 **8.4.2.3 type attribute**

810 The value of the IMPLIED *type* attribute identifies the document type of the external information
811 about the *Party*. It MUST be a URI that defines the namespace associated with the information
812 about the *Party*. If the *type* attribute is omitted, the external information about the *Party* MUST
813 be an HTML web page.

814

815 **8.4.2.4 schemaLocation attribute**

816 The value of the IMPLIED *schemaLocation* attribute provides a URI for the schema that
817 describes the structure of the external information.

818

819 An example of the *PartyRef* element is:

820

```
821     <tp:PartyRef xlink:type="simple"
822               xlink:href="http://example2.com/ourInfo.xml"
823               tp:type="urn:oasis:names:tc:ebxml-cppa:contact-info"
824               tp:schemaLocation="http://example2.com/ourInfo.xsd"/>
```

826

8.4.3 CollaborationRole element

827 The *CollaborationRole* element associates a *Party* with a specific role in the *Business*
828 *Collaboration*. Generally, the *Process-Specification* is defined in terms of roles such as "buyer"
829 and "seller". The association between a specific *Party* and the role(s) it is capable of fulfilling
830 within the context of a *Process-Specification* is defined in both the *CPP* and *CPA* documents. In
831 a *CPP*, the *CollaborationRole* element identifies which role the *Party* is capable of playing in
832 each *Process Specification* documents referenced by the *CPP*. An example of the
833 *CollaborationRole* element, based on RosettaNet™ PIP 3A4 is:

834

```
835     <tp:CollaborationRole >
836       <tp:ProcessSpecification
837         tp:version="2.0a"
838         tp:name="PIP3A4RequestPurchaseOrder"
839         xlink:type="simple"
840         xlink:href="http://www.rosettanet.org/processes/3A4.xml"/>
841       <tp:Role
842         tp:name="Buyer"
843         xlink:type="simple"
844
845         xlink:href="http://www.rosettanet.org/processes/3A4.xml#BuyerId"/>
846       <tp:ApplicationCertificateRef tp:certId="CompanyA_AppCert"/>
847       <tp:ServiceBinding>
848         <tp:Service
849           tp:type="anyURI">urn:icann:rosettanet.org:bpid:3A4$2.0</tp:Service>
850         <tp:CanSend>
851           <tp:ThisPartyActionBinding
852             tp:id="companyA_ABID1"
853             tp:action="Purchase Order Request Action"
854             tp:packageId="CompanyA_RequestPackage">
855             <tp:BusinessTransactionCharacteristics
856               tp:isNonRepudiationRequired="true"
857               tp:isNonRepudiationReceiptRequired="true"
858               tp:isConfidential="transient"
859               tp:isAuthenticated="persistent"
860               tp:isTamperProof="persistent"
861               tp:isAuthorizationRequired="true"
862               tp:timeToAcknowledgeReceipt="PT2H"
863               tp:timeToPerform="P1D"/>
864             <tp>ActionContext
865               tp:binaryCollaboration="Request Purchase Order"
866               tp:businessTransactionActivity="Request Purchase
867               Order"
868               tp:requestOrResponseAction="Purchase Order Request
```



```

869     Action"/>
870         <tp:ChannelId>asyncChannelA1</tp:ChannelId>
871     </tp:ThisPartyActionBinding>
872 </tp:CanSend>
873 <tp:CanSend>
874     <tp:ThisPartyActionBinding
875         tp:id="companyA_ABID2"
876         tp:action="ReceiptAcknowledgment"
877         tp:packageId="CompanyA_ReceiptAcknowledgmentPackage">
878     <tp:BusinessTransactionCharacteristics
879         tp:isNonRepudiationRequired="true"
880         tp:isNonRepudiationReceiptRequired="true"
881         tp:isConfidential="transient"
882         tp:isAuthenticated="persistent"
883         tp:isTamperProof="persistent"
884         tp:isAuthorizationRequired="true"
885         tp:timeToAcknowledgeReceipt="PT2H"
886         tp:timeToPerform="P1D"/>
887     <tp:ChannelId>asyncChannelA1</tp:ChannelId>
888 </tp:ThisPartyActionBinding>
889 </tp:CanSend>
890 <tp:CanReceive>
891     <tp:ThisPartyActionBinding
892         tp:id="companyA_ABID3"
893         tp:action="Purchase Order Confirmation Action"
894         tp:packageId="CompanyA_ResponsePackage">
895     <tp:BusinessTransactionCharacteristics
896         tp:isNonRepudiationRequired="true"
897         tp:isNonRepudiationReceiptRequired="true"
898         tp:isConfidential="transient"
899         tp:isAuthenticated="persistent"
900         tp:isTamperProof="persistent"
901         tp:isAuthorizationRequired="true"
902         tp:timeToAcknowledgeReceipt="PT2H"
903         tp:timeToPerform="P1D"/>
904     <tp>ActionContext
905         tp:binaryCollaboration="Request Purchase Order"
906         tp:businessTransactionActivity="Request Purchase
907 Order"
908         tp:requestOrResponseAction="Purchase Order
909 Confirmation Action"/>
910     <tp:ChannelId>asyncChannelA1</tp:ChannelId>
911 </tp:ThisPartyActionBinding>
912 </tp:CanReceive>
913 <tp:CanReceive>
914     <tp:ThisPartyActionBinding
915         tp:id="companyA_ABID4"
916         tp:action="ReceiptAcknowledgment"
917         tp:packageId="CompanyA_ReceiptAcknowledgmentPackage">
918     <tp:BusinessTransactionCharacteristics
919         tp:isNonRepudiationRequired="true"
920         tp:isNonRepudiationReceiptRequired="true"
921         tp:isConfidential="transient"
922         tp:isAuthenticated="persistent"
923         tp:isTamperProof="persistent"
924         tp:isAuthorizationRequired="true"
925         tp:timeToAcknowledgeReceipt="PT2H"
926         tp:timeToPerform="P1D"/>
927     <tp:ChannelId>asyncChannelA1</tp:ChannelId>
928 </tp:ThisPartyActionBinding>
929 </tp:CanReceive>
930 <tp:CanReceive>
931     <tp:ThisPartyActionBinding

```

```

932         tp:id="companyA_ABID5"
933         tp:action="Exception"
934         tp:packageId="CompanyA_ExceptionPackage">
935         <tp:BusinessTransactionCharacteristics
936             tp:isNonRepudiationRequired="true"
937             tp:isNonRepudiationReceiptRequired="true"
938
939             tp:isConfidential="transient"
940             tp:isAuthenticated="persistent"
941             tp:isTamperProof="persistent"
942             tp:isAuthorizationRequired="true"
943             tp:timeToAcknowledgeReceipt="PT2H"
944             tp:timeToPerform="P1D"/>
945         <tp:ChannelId>asyncChannelA1</tp:ChannelId>
946     </tp:ThisPartyActionBinding>
947 </tp:CanReceive>
948 </tp:ServiceBinding>
949 </tp:CollaborationRole>
950

```

951 To indicate that the *Party* can play roles in more than one *Business Collaboration* or more than
 952 one role in a given *Business Collaboration*, the **PartyInfo** element SHALL contain more than
 953 one **CollaborationRole** element. Each **CollaborationRole** element SHALL contain the
 954 appropriate combination of **ProcessSpecification** element and **Role** element.

955
 956 The **CollaborationRole** element SHALL consist of the following child elements: a REQUIRED
 957 **ProcessSpecification** element, a REQUIRED **Role** element, zero or one
 958 **ApplicationCertificateRef** elements, zero or one **ApplicationSecurityDetailsRef** element, and
 959 one **ServiceBinding** element. The **ProcessSpecification** element identifies the *Process-*
 960 *Specification* document that defines such role. The **Role** element identifies which role the *Party*
 961 is capable of supporting. The **ApplicationCertificateRef** element identifies the certificate to be
 962 used for application level signature and encryption. The **ApplicationSecurityDetailsRef** element
 963 identifies the trust anchors and security policy that will be applied to any application-level
 964 certificate offered by the other *Party*. The **ServiceBinding** element SHALL consist of zero or
 965 more **CanSend** elements and zero or more **CanReceive** elements. The **CanSend** and **CanReceive**
 966 elements identify the **DeliveryChannel** elements that are to be used for sending and receiving
 967 business action messages by the **Role** in question. They MAY also be used for specifying
 968 **DeliveryChannels** for business signal messages.

969
 970 Each *Party* SHALL have a default delivery channel for the delivery of standalone *Message*
 971 Service Handler level signals like (Reliable Messaging) Acknowledgments, Errors,
 972 StatusRequest, StatusResponse, etc.

974 8.4.4 ProcessSpecification element

975 The **ProcessSpecification** element provides the link to the *Process-Specification* document that
 976 defines the interactions between the two *Parties*. It is RECOMMENDED that this *Business-*
 977 *Collaboration* description be prepared in accordance with the ebXML Business Process
 978 Specification Schema[ebBPSS]. The *Process-Specification* document MAY be kept in an
 979 ebXML Registry.

980
 981 NOTE: A *Party* can describe the *Business Collaboration* using any desired alternative to
Collaboration-Protocol Profile and Agreement Specification Page
 26 of 156

982 the ebXML Business Process Specification Schema. When an alternative *Business-*
 983 *Collaboration* description is used, the *Parties* to a *CPA* MUST agree on how to interpret
 984 the *Business-Collaboration* description and how to interpret the elements in the *CPA* that
 985 reference information in the *Business-Collaboration* description. The affected elements
 986 in the *CPA* are the *Role* element, the *CanSend* and *CanReceive* elements, the
 987 *ActionContext* element, and some attributes of the *BusinessTransactionCharacteristics*
 988 element.

989
 990 The syntax of the *ProcessSpecification* element is:

```

991
992     <tp:ProcessSpecification
993         tp:version="2.0a"
994         tp:name="PIP3A4RequestPurchaseOrder"
995         xlink:type="simple"
996         xlink:href="http://www.rosettanet.org/processes/3A4.xml"
997         uuid="urn:icann:rosettanet.org:bpid:3A4$2.0">
998         <ds:Reference ds:URI="http://www.rosettanet.org/processes/3A4.xml">
999             <ds:Transforms>
1000                 <ds:Transform
1001 ds:Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
1002                 </ds:Transforms>
1003                 <ds:DigestMethod
1004                     ds:Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
1005                     <ds:DigestValue>j6lwx3rvEPO0vKtMup4NbeVu8nk=</ds:DigestValue>
1006                 </ds:Reference>
1007         </tp:ProcessSpecification>
1008

```

1009 The *ProcessSpecification* element has zero or more child *ds:Reference* elements, and the
 1010 following attributes:

- 1011 • a REQUIRED *name* attribute,
- 1012 • a REQUIRED *version* attribute,
- 1013 • a FIXED *xlink:type* attribute,
- 1014 • a REQUIRED *xlink:href* attribute,
- 1015 • an IMPLIED *uuid* attribute.

1016
 1017 The *ProcessSpecification* element contains zero or more *ds:Reference* elements formulated
 1018 according to the XML Digital Signature specification[XMLDSIG]. The first *ds:Reference*
 1019 element, if present, relates to the *xlink:type* and *xlink:href* attributes as follows. Each
 1020 *ProcessSpecification* element SHALL contain one *xlink:href* attribute and one *xlink:type*
 1021 attribute with a value of "simple". In case the *CPP* (*CPA*) document is signed, the first
 1022 *ds:Reference* element that is present MUST include a *ds:URI* attribute whose value is identical
 1023 to that of the *xlink:href* attribute in the enclosing *ProcessSpecification* element. The
 1024 *ds:Reference* element specifies a digest method and digest value to enable verification that the
 1025 referenced *Process-Specification* document has not changed. Additional *ds:Reference* elements
 1026 are needed if the referenced *ProcessSpecification* in turn includes (i.e., references) other
 1027 *ProcessSpecifications*. Essentially, *ds:Reference* elements MUST be provided to correspond to
 1028 the transitive closure of all *ProcessSpecifications* that are referenced directly or indirectly to
 1029 ensure that none of them has been changed.
 1030

1031 **8.4.4.1 name attribute**

1032 The *ProcessSpecification* element MUST include a REQUIRED *name* attribute: a string that
1033 identifies the *Business Process-Specification* being performed. If the *Process-Specification*
1034 document is defined by the ebXML Business Process specification [ebBPSS], then this attribute
1035 MUST be set to the *name* for the corresponding *ProcessSpecification* element within the
1036 *Business Process Specification* instance.

1037

1038 **8.4.4.2 version attribute**

1039 The *ProcessSpecification* element includes a REQUIRED *version* attribute to indicate the
1040 version of the *Process-Specification* document identified by the *xlink:href* attribute (and also
1041 identified by the *ds:Reference* element, if any).

1042

1043 **8.4.4.3 xlink:type attribute**

1044 The *xlink:type* attribute has a FIXED value of "simple". This identifies the element as being an
1045 [XLINK] simple link.

1046

1047 **8.4.4.4 xlink:href attribute**

1048 The REQUIRED *xlink:href* attribute SHALL have a value that identifies the *Process-*
1049 *Specification* document and is a URI that conforms to [RFC2396].

1050

1051 **8.4.4.5 uuid attribute**

1052 The IMPLIED *uuid* attribute uniquely identifies the *ProcessSpecification*. If the *Process-*
1053 *Specification* document is defined by the ebXML Business Process specification [ebBPSS], then
1054 this attribute MUST be set to the *uuid* for the corresponding *ProcessSpecification* element
1055 within the business process specification instance.

1056

1057 **8.4.4.6 ds:Reference element**

1058 The *ds:Reference* element identifies the same *Process-Specification* document as the enclosing
1059 *ProcessSpecification* element's *xlink:href* attribute and additionally provides for verification that
1060 the *Process-Specification* document has not changed since the *CPP* was created, through the use
1061 of a digest method and digest value as described below.

1062

1063 NOTE: *Parties* MAY test the validity of the *CPP* or *CPA* at any time. The following
1064 validity tests MAY be of particular interest:

1065

- 1066 • test of the validity of a *CPP* and the referenced *Process-Specification* documents at
1067 the time composition of a *CPA* begins in case they have changed since they were
1068 created,
- 1069 • test of the validity of a *CPA* and the referenced *Process-Specification* documents at
1070 the time a *CPA* is installed into a *Party's* system,
- 1071 • test of the validity of a *CPA* at intervals after the *CPA* has been installed into a *Party's*
1072 system. The *CPA* and the referenced *Process-Specification* documents MAY be
1073 processed by an installation tool into a form suited to the particular middleware.
1074 Therefore, alterations to the *CPA* and the referenced *Process-Specification* documents
1075 do not necessarily affect ongoing run-time operations. Such alterations might not be

1076 detected until it becomes necessary to reinstall the *CPA* and the referenced *Process-*
 1077 *Specification* documents.

1078
 1079 The syntax and semantics of the *ds:Reference* element and its child elements are defined in the
 1080 XML Digital Signature specification[XMLDSIG]. In addition, to identify the *Process-*
 1081 *Specification* document, the first *ds:Reference* element MUST include a *ds:URI* attribute whose
 1082 value is identical to that of the *xlink:href* attribute in the enclosing *ProcessSpecification*
 1083 element.

1084
 1085 According to [XMLDSIG], a *ds:Reference* element can have a *ds:Transforms* child element,
 1086 which in turn has an ordered list of one or more *ds:Transform* child elements to specify a
 1087 sequence of transforms. However, this specification currently REQUIRES the Canonical
 1088 XML[XMLC14N] transform and forbids other transforms. Therefore, the following additional
 1089 requirements apply to a *ds:Reference* element within a *ProcessSpecification* element:

- 1090
- 1091 • The *ds:Reference* element MUST have a *ds:Transforms* child element.
 - 1092 • That *ds:Transforms* element MUST have exactly one *ds:Transform* child element.
 - 1093 • That *ds:Transform* element MUST specify the Canonical XML[XMLC14N] transform
 1094 via the following REQUIRED value for its REQUIRED *ds:Algorithm* attribute:
 1095 <http://www.w3.org/TR/2001/Rec-xml-c14n-20010315>.

1096
 1097 Note that implementation of Canonical XML is REQUIRED by the XML Digital
 1098 Signature specification[XMLDSIG].
 1099

1100 To enable verification that the identified and transformed *Process-Specification* document has
 1101 not changed, the *ds:DigestMethod* element specifies the digest algorithm applied to the *Process-*
 1102 *Specification* document, and the *ds:DigestValue* element specifies the expected value. The
 1103 *Process-Specification* document is presumed to be unchanged if and only if the result of applying
 1104 the digest algorithm to the *Process-Specification* document results in the expected value.
 1105

1106 A *ds:Reference* element in a *ProcessSpecification* element has implications for *CPP* validity:

- 1107
- 1108 • A *CPP* MUST be considered invalid if any *ds:Reference* element within a
 1109 *ProcessSpecification* element fails reference validation as defined by the XML Digital
 1110 Signature specification[XMLDSIG].
 - 1111 • A *CPP* MUST be considered invalid if any *ds:Reference* element within it cannot be
 1112 dereferenced.

1113
 1114
 1115 Other validity implications of such *ds:Reference* elements are specified in the description of the
 1116 *Signature* element in Section 9.9.
 1117

1118 NOTE: The XML Digital Signature specification[XMLDSIG] states "The signature
 1119 application MAY rely upon the identification (URI) and Transforms provided by the
 1120 signer in the Reference element, or it MAY obtain the content through other means such

1121 as a local cache" (emphasis on MAY added). However, it is RECOMMENDED that
 1122 ebXML *CPP/CPA* implementations not make use of such cached results when signing or
 1123 validating.

1124
 1125 NOTE: It is recognized that the XML Digital Signature specification[XMLDSIG]
 1126 provides for signing an XML document together with externally referenced documents.
 1127 In cases where a *CPP* or *CPA* document is in fact suitably signed, that facility could also
 1128 be used to ensure that the referenced *Process-Specification* documents are unchanged.
 1129 However, this specification does not currently mandate that a *CPP* or *CPA* be signed.

1130
 1131 NOTE: If the *Parties* to a *CPA* wish to customize a previously existing *Process-*
 1132 *Specification* document, they MAY copy the existing document, modify it, and cause
 1133 their *CPA* to reference the modified copy. It is recognized that for reasons of clarity,
 1134 brevity, or historical record, the *Parties* might prefer to reference a previously existing
 1135 *Process-Specification* document in its original form and accompany that reference with a
 1136 specification of the agreed modifications. Therefore, *CPP* usage of the ***ds:Reference***
 1137 element's ***ds:Transforms*** sub-element within a ***ProcessSpecification*** element might be
 1138 expanded in the future to allow other transforms as specified in the XML Digital
 1139 Signature specification[XMLDSIG]. For example, modifications to the original
 1140 document could then be expressed as XSLT transforms. After applying any transforms,
 1141 it would be necessary to validate the transformed document against the ebXML Business
 1142 Process Specification Schema[ebBPSS].

1143

1144 **8.4.5 Role element**

1145 The REQUIRED ***Role*** element identifies which role in the *Process Specification* the *Party* is
 1146 capable of supporting via the ***ServiceBinding*** element(s) siblings within this ***CollaborationRole***
 1147 element.

1148

1149 The ***Role*** element has the following attributes:

- 1150 • a REQUIRED ***name*** attribute,
- 1151 • a FIXED ***xlink:type*** attribute,
- 1152 • a REQUIRED ***xlink:href*** attribute.

1153

1154 **8.4.5.1 name attribute**

1155 The REQUIRED ***name*** attribute is a string that gives a name to the ***Role***. Its value is
 1156 taken from a name attribute of one of a ***BinaryCollaboration***'s ***Role*** elements described in the
 1157 *Process Specification*[ebBPSS].

1158

1159 See NOTE in Section 8.4.4 regarding alternative *Business-Collaboration* descriptions.

1160

1161 **8.4.5.2 xlink:type attribute**

1162 The ***xlink:type*** attribute has a FIXED value of "simple". This identifies the element as being an
 1163 [XLINK] simple link.

1164

1165 **8.4.5.3 xlink:href attribute**

1166 The REQUIRED *xlink:href* attribute SHALL have a value that is a URI that conforms to
1167 [RFC2396]. It identifies the location of the element or attribute within the *Process-Specification*
1168 document that defines the role in the context of the *Business Collaboration*. An example is:

```
1169         xlink:href="http://www.rosettanet.org/processes/3A4.xml#Buyer"  
1170
```

1171
1172 Where "Buyer" is the value of the ID attribute of the element in the *Process-Specification*
1173 document that defines the role name.

1174

1175 **8.4.6 ApplicationCertificateRef element**

1176 The *ApplicationCertificateRef* element, if present, identifies a certificate for use by the business
1177 process/application layer. This certificate is not used by the ebXML messaging system, but it is
1178 included in the *CPP* so that it can be considered in the *CPA* negotiation process. The
1179 *ApplicationCertificateRef* element can occur zero or more times.

1180 NOTE: It is up to the application software on both sides of a collaboration to determine
1181 the intended/allowed usage of an application certificate by inspecting the key usage
1182 extension within the certificate itself.

1183 NOTE: This element is included in the *CPP/CPA* to support interoperability with legacy
1184 systems that already perform cryptographic functions such as digital signature or
1185 encryption. Implementers should understand that use of *ApplicationCertificateRef* is
1186 necessary only in cases where interoperability with such legacy systems is required.

1187
1188 The *ApplicationCertificateRef* element has

- 1189 • A REQUIRED *certId* attribute.

1190

1191 **8.4.6.1 certId attribute**

1192 The REQUIRED *certId* attribute is an [XML] IDREF that associates the *CollaborationRole*
1193 element with a certificate. It MUST have a value equal the value of the *certId* attribute of one of
1194 the *Certificate* elements under *PartyInfo*.

1195

1196 **8.4.7 ApplicationSecurityDetailsRef element**

1197 The *ApplicationSecurityDetailsRef* element, if present, identifies the trust anchors and security
1198 policy that this *Party* will apply to any application-level certificate offered by the other *Party*.
1199 These trust anchors and policy are not used by the ebXML messaging system, but are included in
1200 the *CPP* so that they can be considered in the *CPA* negotiation process.

1201

1202 The *ApplicationSecurityDetailsRef* element has

- 1203 • A REQUIRED *securityId* attribute.

1204 •

1205 **8.4.7.1 SecurityId attribute**

1206 The REQUIRED *securityId* attribute is an [XML] IDREF that associates the *CollaborationRole*
1207 with a *SecurityDetails* element that specifies a set of trust anchors and a security policy. It
1208 MUST have a value equal to the value of the *securityId* attribute of one of the *SecurityDetails*
1209 elements under *PartyInfo*.

1210

1211 **8.4.8 ServiceBinding element**

1212 The *ServiceBinding* element identifies a *DeliveryChannel* element for all of the business
1213 *Message* traffic that is to be sent or received by the *Party* within the context of the identified
1214 *Process-Specification* document. It MUST contain at least one *CanReceive* or *CanSend* child
1215 element.

1216

1217 The *ServiceBinding* element has one child *Service* element, zero or more *CanSend* child
1218 elements, and zero or more *CanReceive* child elements.

1219

1220 **8.4.9 Service element**

1221 The value of the *Service* element is a string that SHALL be used as the value of the *Service*
1222 element in the ebXML *Message Header*[ebMS] or a similar element in the *Message Header* of
1223 an alternative *message* service. The *Service* element has an IMPLIED *type* attribute.

1224

1225 If the *Process-Specification* document is defined by the ebXML Business Process Specification
1226 Schema[ebBPSS], then the value of the *Service* element MUST be the *uuid* (URI) attribute
1227 specified for the *ProcessSpecification* element in the Business Process Specification Schema
1228 instance document.

1229

1230 NOTE: The purpose of the *Service* element is to provide routing information for the
1231 ebXML *Message Header*. The *CollaborationRole* element and its child elements identify
1232 the information in the *ProcessSpecification* document that is relevant to the *CPP* or *CPA*.
1233 The *Service* element MAY be used along with the *CanSend* and *CanReceive* elements
1234 (and their descendants) to provide routing of received messages to the correct application
1235 entry point.

1236

1237 **8.4.9.1 type attribute**

1238 If the *type* attribute is present, it indicates that the *Parties* sending and receiving the *Message*
1239 know, by some other means, how to interpret the value of the *Service* element. The two *Parties*
1240 MAY use the value of the *type* attribute to assist the interpretation.

1241

1242 If the *type* attribute is not present, the value of the *Service* element MUST be a URI[RFC2396].
1243 If using the ebXML Business Process Specification[ebBPSS] for defining the *Process-*
1244 *Specification* document, the *type* attribute MUST be a URI[RFC2396].

1245

1246 **8.4.10 CanSend element**

1247 The *CanSend* element identifies an *action* message that a *Party* is capable of sending. It has
Collaboration-Protocol Profile and Agreement Specification

Page

1248 three sub-elements: ***ThisPartyActionBinding***, ***OtherPartyActionBinding***, and ***CanReceive***. The
1249 ***ThisPartyActionBinding*** element is REQUIRED for both *CPPs* and *CPAs*. It identifies the
1250 ***DeliveryChannel*** and the ***Packaging*** the *Party* described by the encompassing ***PartyInfo***
1251 element will use for sending the *action* invocation message in question. The
1252 ***OtherPartyActionBinding*** element is only used in the case of *CPAs*. Within a *CPA* and under the
1253 same ***CanSend*** element, the ***DeliveryChannels*** and ***Packaging*** used/expected by the two *Parties*
1254 MUST be compatible. The ***CanReceive*** element can occur zero or more times. When present, it
1255 indicates that one or more synchronous response actions are expected.
1256 This is illustrated in the *CPP* and *CPA* examples in the appendices.

1257
1258
1259 NOTE: While the schema permits arbitrary nesting levels under the ***CanSend*** element,
1260 use cases for nesting beyond two levels have not yet been presented. Two levels could be
1261 needed for a Request with a synchronously returned Response that additionally specified
1262 a synchronously returned Acknowledgment for that Response.

1263
1264

1265 8.4.11 CanReceive element

1266 The ***CanReceive*** element identifies an *action* invocation message that a *Party* is capable of
1267 receiving. It has three sub-elements: ***ThisPartyActionBinding***, ***OtherPartyActionBinding***, and
1268 ***CanSend***. The ***ThisPartyActionBinding*** element is REQUIRED for both *CPPs* and *CPAs*. It
1269 identifies the ***DeliveryChannel*** the *Party* described by the encompassing ***PartyInfo*** element will
1270 use for receiving the *action* message in question and the ***Packaging*** it is expecting. The
1271 ***OtherPartyActionBinding*** element is only used in the case of *CPAs*. Within a *CPA* and under
1272 the same ***CanReceive*** element, the ***DeliveryChannels*** and ***Packaging*** used/expected by the two
1273 *Parties* MUST be compatible. The ***CanSend*** element can occur zero or more times. When
1274 present, it indicates that one or more synchronous response actions are expected. This is
1275 illustrated in the *CPP* and *CPA* examples in the appendices.

1276
1277 NOTE: While the schema permits arbitrary nesting levels under the ***CanReceive*** element,
1278 use cases for nesting beyond two levels have not yet been presented. Two levels could be
1279 needed for a Request with a synchronously returned Response that additionally specified
1280 a synchronously returned Acknowledgment for that Response.

1281

1282 8.4.12 ThisPartyActionBinding element

1283 The ***ThisPartyActionBinding*** specifies one or more ***DeliveryChannel*** elements for *Messages* for
1284 a selected *action* and the ***Packaging*** for those *Messages* that are to be sent or received by the
1285 *Party* in the context of the *Process Specification* that is associated with the parent
1286 ***ServiceBinding*** element.

1287

1288 The ***ThisPartyActionBinding*** element has a REQUIRED child
1289 ***BusinessTransactionCharacteristics*** element, zero or one child ***ActionContext*** element and one
1290 or more ***ChannelID*** child elements.

1291
 1292 The ***ThisPartyActionBinding*** element has the following attributes:

- 1293 • a REQUIRED ***action*** attribute,
- 1294 • a REQUIRED ***packageId*** attribute,
- 1295 • an IMPLIED ***xlink:href*** attribute,
- 1296 • a FIXED ***xlink:type*** attribute.

1297
 1298 Under a given ***ServiceBinding*** element, there MAY be multiple ***CanSend*** or ***CanReceive*** child
 1299 elements with the same ***action*** to allow different software entry points and Transport options. In
 1300 such a scenario, the ***DeliveryChannels*** referred by the ***ChannelID*** child elements of
 1301 ***ThisPartyActionBinding*** SHALL point to distinct ***EndPoints*** for the receiving MSH to uniquely
 1302 identify the ***DeliveryChannel*** being used for this particular message exchange.

1303
 1304 NOTE: An implementation MAY provide the capability of dynamically assigning
 1305 delivery channels on a per ***Message*** basis during performance of the
 1306 ***BinaryCollaboration***. The delivery channel selected would be chosen, based on present
 1307 conditions, from those identified by ***CanSend*** elements that refer to the
 1308 ***BinaryCollaboration*** that is sending the ***Message***. On the receiving side, the MSH can
 1309 use the distinct ***EndPoints*** to identify the ***DeliveryChannel*** used for this message
 1310 exchange.

1311
 1312 Within a ***CanSend*** element or a ***CanReceive*** element, when both the ***ThisPartyActionBinding***
 1313 and ***OtherPartyActionBinding*** elements are present (i.e., in a ***CPA***), they MUST have identical
 1314 action values or equivalent ***ActionContext*** elements. In addition, the ***DeliveryChannel*** and
 1315 ***Packaging*** that they reference MUST be compatible.

1316 1317 **8.4.12.1 action attribute**

1318 The value of the REQUIRED ***action*** attribute is a string that identifies the business document
 1319 exchange to be associated with the ***DeliveryChannel*** identified by the ***ChannelId*** sub-elements.
 1320 The value of the ***action*** attribute SHALL be used as the value of the ***Action*** element in the
 1321 ebXML ***Message Header***[ebMS] or a similar element in the ***Message Header*** of an alternative
 1322 ***message*** service. The purpose of the ***action*** attribute is to provide a mapping between the
 1323 hierarchical naming associated with a ***Business Process/Application*** and the ***Action*** element in
 1324 the ebXML ***Message Header***[ebMS]. This mapping MAY be implemented by using the
 1325 ***ActionContext*** element. See NOTE in Section 8.4.4 regarding alternative ***Business***
 1326 ***Collaboration*** descriptions.

1327
 1328 Business signals, when sent individually (i.e., not bundled with response documents in
 1329 synchronous reply mode), SHALL use the values ***ReceiptAcknowledgment***,
 1330 ***AcceptanceAcknowledgment***, or ***Exception*** as the value of their ***action*** attribute. In addition, they
 1331 SHOULD specify a ***Service*** that is the same as the ***Service*** used for the original message.

1332
 1333 NOTE: In general, the action name chosen by the two ***Parties*** to represent a particular
 1334 requesting business activity or responding business activity in the context of a ***Binary***
 1335 ***Collaboration*** that makes use of nested ***BinaryCollaborations*** MAY not be identical.

1336 Therefore, when composing two *CPPs* to form a *CPA*, it is necessary to make use of
1337 information from the associated *ActionContext* (see Section 8.4.16) in order to determine
1338 if two different action names from the two *CPPs* actually represent the same
1339 *ActionContext*. When business transactions are not reused in different contexts, it is
1340 recommended that the names of the requesting business activity and responding business
1341 activity be used as action names.

1342

1343 **8.4.12.2 packageId attribute**

1344 The REQUIRED *packageId* attribute is an [XML] IDREF that identifies the *Packaging* element
1345 to be associated with the *Message* identified by the *action* attribute.

1346

1347 **8.4.12.3 xlink:href attribute**

1348 The IMPLIED *xlink:href* attribute, if present, SHALL provide an absolute [XPOINTER] URI
1349 expression that specifically identifies the *RequestingBusinessActivity* or
1350 *RespondingBusinessActivity* element within the associated *ProcessSpecification*
1351 document[*ebBPSS*] that is identified by the *ProcessSpecification* element.

1352

1353 **8.4.12.4 xlink:type attribute**

1354 The IMPLIED *xlink:type* attribute has a FIXED value of "simple". This identifies the element as
1355 being an [XLINK] simple link.

1356

1357 **8.4.13 OtherPartyActionBinding**

1358 The *OtherPartyActionBinding* element is only used in the case of *CPAs*. It is of type IDREF and
1359 identifies a matching *ThisPartyActionBinding* element that is found under the collaboration
1360 partner's *PartyInfo*. It indirectly identifies the *DeliveryChannel* the other *Party* will use for
1361 sending or receiving the *action* message in question and the expected *Packaging*. Within a *CPA*
1362 and under the same *CanSend* or *CanReceive* element, the *DeliveryChannels* and *Packaging*
1363 used/expected by the two *Parties*, as indicated by the *ThisPartyActionBinding* and
1364 *OtherPartyActionBinding* elements, MUST be compatible.

1365

1366 **8.4.14 BusinessTransactionCharacteristics element**

1367 The *BusinessTransactionCharacteristics* element describes the security characteristics and other
1368 attributes of the delivery channel, as derived from the *ProcessSpecification(s)* whose messages
1369 are transported using the delivery channel. The attributes of the
1370 *BusinessTransactionCharacteristics* element, MAY be used to override the values of the
1371 corresponding attributes in the *ProcessSpecification* document.

1372

1373 See NOTE in Section 8.4.4 regarding alternative *Business-Collaboration* descriptions.

1374

1375 *CPP* and *CPA* composition tools and *CPA* deployment tools SHALL check the delivery channel
1376 definitions for the sender and receiver (transport and document-exchange) for internal
1377 consistency as well as compatibility between the two partners. Typically, when an attribute has a
1378 particular value, sub-elements under the corresponding Transport and DocExchange elements

1379 would exist to further describe the implied implementation parameters.

1380

1381 The ***BusinessTransactionCharacteristics*** element has the following attributes:

1382

- 1383 • an IMPLIED ***isNonRepudiationRequired*** attribute,
- 1384 • an IMPLIED ***isNonRepudiationReceiptRequired*** attribute,
- 1385 • an IMPLIED ***isConfidential*** attribute,
- 1386 • an IMPLIED ***isAuthenticated*** attribute,
- 1387 • an IMPLIED ***isAuthorizationRequired*** attribute,
- 1388 • an IMPLIED ***isTamperProof*** attribute,
- 1389 • an IMPLIED ***isIntelligibleCheckRequired*** attribute,
- 1390 • an IMPLIED ***timeToAcknowledgeReceipt*** attribute,
- 1391 • an IMPLIED ***timeToAcknowledgeAcceptance*** attribute,
- 1392 • an IMPLIED ***timeToPerform*** attribute,
- 1393 • an IMPLIED ***retryCount*** attribute.

1394

1395 These attributes allow parameters specified at the *Process-Specification* level to be overridden. If
1396 one of these attributes is not specified, the corresponding default value should be obtained from
1397 the *Process-Specification* document.

1398

1399 **8.4.14.1 isNonRepudiationRequired attribute**

1400 The ***isNonRepudiationRequired*** attribute is a Boolean with possible values of "true" and
1401 "false". If the value is "true" then the delivery channel MUST specify that the *Message* is to be
1402 digitally signed using the certificate of the *Party* sending the *Message*, and archived by both
1403 *Parties*. The ***SenderNonRepudiation*** element under ***DocExchange/ebXMLSenderBinding*** (see
1404 Section 8.4.43) and the ***ReceiverNonRepudiation*** element under
1405 ***DocExchange/ebXMLReceiverBinding*** (see Section 0) further describe various parameters
1406 related to the implementation of non-repudiation of origin, such as the hashing algorithm, the
1407 signature algorithm, the signing certificate, the trust anchor, etc.

1408

1409 **8.4.14.2 isNonRepudiationReceiptRequired attribute**

1410 The ***isNonRepudiationReceiptRequired*** attribute is a Boolean with possible values of "true"
1411 and "false". If the value is "true" then the delivery channel MUST specify that the *Message* is to
1412 be acknowledged by a digitally signed *Receipt Acknowledgment* signal *Message*, signed using
1413 the certificate of the *Party* that received the *Message*, that includes the digest(s) of the payload(s)
1414 of the *Message* being acknowledged. The ***SenderNonRepudiation*** element under
1415 ***DocExchange/ebXMLSenderBinding*** (see Section 8.4.43) and the ***ReceiverNonRepudiation***
1416 element under ***DocExchange/ebXMLReceiverBinding*** (see Section 0) further describe various
1417 parameters related to the implementation of non-repudiation of receipt.

1418

1419

1420 **8.4.14.3 isConfidential attribute**

1421 The ***isConfidential*** attribute has the possible values of "none", "transient", "persistent", and
1422 "transient-and-persistent". These values MUST be interpreted as defined by the ebXML Business
1423 Process Specification Schema[ebBPSS]. In general, transient confidentiality can be implemented

Collaboration-Protocol Profile and Agreement Specification

Page

1424 using a secure transport protocol like SSL; persistent confidentiality can be implemented using a
1425 digital envelope mechanism like S/MIME. Secure transport information is further provided in the
1426 *TransportSender* (see Section 8.4.25) and *TransportReceiver* (see Section 8.4.32) elements
1427 under the *Transport* element. Persistent encryption information is further provided in the
1428 *SenderDigitalEnvelope* element under *DocExchange/ebXMLSenderBinding* (see Section
1429 8.4.48) and the *ReceiverDigitalEnvelope* element under *DocExchange/ebXMLReceiverBinding*
1430 (see Section 8.4.56).

1431

1432 **8.4.14.4 isAuthenticated attribute**

1433 The *isAuthenticated* attribute has the possible values of "none", "transient", "persistent", and
1434 "persistent-and-transient". If this attribute is set to any value other than "none", then the receiver
1435 MUST be able to verify the identity of the sender. In general, transient authentication can be
1436 implemented using a secure transport protocol like SSL (with or without the use of basic or
1437 digest authentication); persistent authentication can be implemented using a digital signature
1438 mechanism. Secure transport information is further provided in the *TransportSender* (see
1439 Section 8.4.25) and *TransportReceiver* (see Section 8.4.33) elements under the *Transport*
1440 element. Persistent authentication information is further provided in the *SenderNonRepudiation*
1441 element under *DocExchange/ebXMLSenderBinding* (see Section 8.4.43) and the
1442 *ReceiverNonRepudiation* element (under *DocExchange/ebXMLReceiverBinding* (see Section
1443 0)).

1444

1445

1446 **8.4.14.5 isAuthorizationRequired attribute**

1447 The *isAuthorizationRequired* attribute is a Boolean with possible values of "true" and
1448 "false". If the value is "true" then it indicates that the delivery channel MUST specify that the
1449 sender of the *Message* is to be authorized before delivery to the application.

1450

1451 **8.4.14.6 isTamperProof attribute**

1452 The *isTamperProof* attribute has the possible values of "none", "transient", "persistent", and
1453 "persistent-and-transient". If this attribute is set to a value other than "none", then it must be
1454 possible for the receiver to detect if the received message has been corrupted or tampered with.
1455 In general, transient tamper detection can be implemented using a secure transport like SSL;
1456 persistent tamper detection can be implemented using a digital signature mechanism. Secure
1457 transport information is further provided in the *TransportSender* (see Section 8.4.25) and
1458 *TransportReceiver* (see Section 8.4.48) elements under the *Transport* element. Digital signature
1459 information is further provided in the *SenderNonRepudiation* element under
1460 *DocExchange/ebXMLSenderBinding* (see Section 8.4.43) and the *ReceiverNonRepudiation*
1461 element under *DocExchange/ebXMLReceiverBinding* (see Section 0).

1462

1463

1464 **8.4.14.7 isIntelligibleCheckRequired attribute**

1465 The *isIntelligibleCheckRequired* attribute is a Boolean with possible values of "true" and
1466 "false". If the value is "true", then the receiver MUST verify that a business document is not
1467 garbled (i.e., passes schema validation) before returning a *Receipt Acknowledgment* signal.

1468

1469 **8.4.14.8 timeToAcknowledgeReceipt attribute**

1470 The *timeToAcknowledgeReceipt* attribute is of type duration [XMLSCHEMA-2]. It specifies the
1471 time period within which the receiving *Party* has to acknowledge receipt of a business document.

1472
1473 If this attribute is specified, then the *Receipt Acknowledgment* signal MUST be used.

1474
1475 **8.4.14.9 timeToAcknowledgeAcceptance attribute**

1476 The *timeToAcknowledgeAcceptance* attribute is of type duration [XMLSCHEMA-2]. It
1477 specifies the time period within which the receiving *Party* has to non-substantively acknowledge
1478 acceptance of a business document (i.e., after it has passed business rules validation).

1479 If this attribute is specified, then the *Acceptance Acknowledgment* signal MUST be used.

1480
1481 **8.4.14.10 timeToPerform attribute**

1482 The *timeToPerform* attribute is of type duration [XMLSCHEMA-2]. It specifies the time period,
1483 starting from the initiation of the *RequestingBusinessActivity*, within which the initiator of the
1484 transaction MUST have received the response, i.e., the business document associated with the
1485 *RespondingBusinessActivity*.

1486
1487 NOTE: The *timeToPerform* attribute associated with a *BinaryCollaboration* in BPSS is
1488 currently not modeled in this specification. Therefore, it cannot be overridden. In other
1489 words, the value specified at the BPSS level MUST be used.

1490
1491 When synchronous reply mode is in use (see Section 8.4.23.1), the *TimeToPerform* value
1492 SHOULD be used as the connection timeout.

1493
1494 **8.4.14.11 retryCount attribute**

1495 The *retryCount* attribute is of type integer. It specifies the maximum number of times the
1496 *Business Transaction* MAY be retried should certain error conditions (e.g., time out waiting for
1497 the Receipt Acknowledgment signal) arise during its execution. Such retries MUST not be used
1498 when ebXML Reliable Messaging is employed to transport messages in the *Business*
1499 *Transaction*. In the latter case, retries are governed by the *Retry*, *RetryInterval* elements under
1500 the *ReliableMessaging* element.

1501
1502 **8.4.15 ChannelId element**

1503 The *ChannelId* element identifies one or more *DeliveryChannel* elements that can be used for
1504 sending or receiving the corresponding action messages. Multiple *ChannelId* elements can be
1505 used to associate *DeliveryChannel* elements with different characteristics with the same
1506 *CanSend* or *CanReceive* element. For example, a *Party* that supports both HTTP and SMTP for
1507 sending the same action can specify different *ChannelId* attribute values for the corresponding
1508 channels. If using multiple *DeliveryChannel* elements, different *EndPoint* elements MUST be
1509 used, so that the receiving MSH can uniquely determine the *DeliveryChannel* element being
1510 used for this message exchange.

1511

1512 8.4.16 ActionContext element

1513 The *ActionContext* element provides a mapping from the *action* attribute in the
1514 *ThisPartyActionBinding* element to the corresponding *Business Process* implementation-
1515 specific naming strategy, if any. If the *Process-Specification* document is defined by the ebXML
1516 Business Process Specification Schema[ebBPSS], the *ActionContext* element MUST be present.

1517
1518 Any business process/application implementation can use a combination of information in the
1519 *action* attribute and the *ActionContext* elements to make message routing decisions. If using
1520 alternative *Business-Collaboration* description schemas, the *action* attribute of the parent
1521 *ThisPartyActionBinding* element and/or the [XMLSCHEMA-1] *wildcard* element within the
1522 *ActionContext* element MAY be used to make routing decisions above the level of the *Message*
1523 Service Handler.

1524
1525 The *ActionContext* element has the following elements:

- 1526 • zero or one CollaborationActivity element,
- 1527 • zero or more [XML SCHEMA-1] *wildcard* elements.

1528
1529 The *ActionContext* element also has the following attributes:

- 1530 • a REQUIRED *binaryCollaboration* attribute,
- 1531 • a REQUIRED businessTransactionActivity attribute,
- 1532 • a REQUIRED requestOrResponseAction attribute.

1533

1534 8.4.16.1 binaryCollaboration attribute

1535 The REQUIRED *binaryCollaboration* attribute is a string that identifies the
1536 *BinaryCollaboration* for which the parent *ThisPartyActionBinding* is defined. If the *Process-*
1537 *Specification* document is defined by the ebXML Business Process Specification
1538 Schema[ebBPSS], then the value of the *binaryCollaboration* attribute MUST match the value of
1539 the *name* attribute of the *BinaryCollaboration* element as defined in the ebXML Business
1540 Process Specification Schema[ebBPSS].

1541

1542 8.4.16.2 businessTransactionActivity attribute

1543 The REQUIRED *businessTransactionActivity* attribute is a string that identifies the *Business*
1544 *Transaction* for which the parent *ThisPartyActionBinding* is defined. If the *Process-*
1545 *Specification* document is defined by the ebXML Business Process Specification
1546 Schema[ebBPSS], the value of the *businessTransactionActivity* attribute MUST match the value
1547 of the *name* attribute of the *BusinessTransactionActivity* element, whose parent is the
1548 *BinaryCollaboration* referred to by the *binaryCollaboration* attribute.

1549

1550 8.4.16.3 requestOrResponseAction attribute

1551 The REQUIRED *requestOrResponseAction* attribute is a string that identifies either the
1552 *Requesting or Responding Business Activity* for which the parent *ThisPartyActionBinding* is
1553 defined. For a *ThisPartyActionBinding* defined for the request side of a message exchange, if
1554 the *Process-Specification* document is defined by the ebXML Business Process Specification
1555 Schema [ebBPSS], the value of the *requestOrResponseAction* attribute MUST match the value

1556 of the *name* attribute of the *RequestingBusinessActivity* element corresponding to the *Business*
 1557 *Transaction* specified in the *businessTransactionActivity* attribute. Similarly, for the response
 1558 side of a message exchange, the value of the *requestOrResponseAction* attribute MUST match
 1559 the value of the *name* attribute of the *RespondingBusinessActivity* element corresponding to the
 1560 *Business Transaction* specified in the *businessTransactionActivity* attribute, as defined in the
 1561 ebXML Business Process Specification Schema[ebBPSS].
 1562

1563 **8.4.17 CollaborationActivity element**

1564 The *CollaborationActivity* element supports the *ActionContext* element by providing the ability
 1565 to map any nested *BinaryCollaborations* as defined in the ebXML Business Process
 1566 Specification Schema[ebBPSS] to the *action* attribute. The *CollaborationActivity* element
 1567 MUST be present when the *BinaryCollaboration* referred to by the *binaryCollaboration*
 1568 attribute has a *CollaborationActivity* defined in the business process definition.
 1569

1570 An example of the *CollaborationActivity* element is:

```
1571
1572     <tp:CollaborationActivity
1573         tp:name="Credit Check"/>
```

1574
 1575 The *CollaborationActivity* element has zero or one child *CollaborationActivity* element to
 1576 indicate further nesting of *BinaryCollaborations*.
 1577

1578 The *CollaborationActivity* element also has one attribute:

- 1579 • a REQUIRED *name* attribute.

1581 **8.4.17.1 name attribute**

1582 The REQUIRED *name* attribute is a string that identifies the *CollaborationActivity* included in
 1583 the *BinaryCollaboration*. If the *Process-Specification* document is defined by the ebXML
 1584 Business Process Specification Schema[ebBPSS], the value of the *name* attribute MUST match
 1585 the value of the *name* attribute of the *CollaborationActivity* within the *BinaryCollaboration*, as
 1586 defined in the ebXML Business Process Specification Schema[ebBPSS].
 1587

1588 **8.4.18 Certificate element**

1589 The *Certificate* element defines certificate information for use in this *CPP*. One or more
 1590 *Certificate* elements can be provided for use in the various security functions in the *CPP*. An
 1591 example of the *Certificate* element is:

```
1592
1593     <tp:Certificate tp:certId="CompanyA_SigningCert">
1594         <ds:KeyInfo> . . . </ds:KeyInfo>
1595     </tp:Certificate>
```

1596
 1597 The *Certificate* element has a single REQUIRED attribute: *certId*. The *Certificate* element has a
 1598 single child element: *ds:KeyInfo*.
 1599

1600 The ***ds:KeyInfo*** element may contain a complete chain of certificates, but the leaf certificate is
 1601 the ***Certificate*** element containing the key used in various asymmetric cryptographic operations.
 1602 (The leaf certificate will be one that has been issued but has not been used to issue certificates.)
 1603 If the leaf certificate has been issued by an intermediate certificate authority, the complete chain
 1604 to the root certificate authority SHOULD be included because it aids in testing certificate
 1605 validity with respect to a set of trust anchors.

1606

1607 **8.4.18.1 certId attribute**

1608 The REQUIRED ***certId*** attribute is an [XML] ID that is referred to by a ***CertificateRef*** element
 1609 elsewhere in the *CPP*. Here is an example of how a ***CertificateRef*** would refer to the ***Certificate***
 1610 element shown in the previous section:

```
1611
1612     <tp:SigningCertificateRef tp:certId="CompanyA_SigningCert" />
1613
```

1613

1614 **8.4.18.2 ds:KeyInfo element**

1615 The ***ds:KeyInfo*** element defines the certificate information. The content of this element and any
 1616 sub-elements are defined by the XML Digital Signature specification[XMLDSIG].

1617

1618 NOTE: Software for creation of *CPPs* and *CPAs* MUST recognize the ***ds:KeyInfo***
 1619 element and insert the sub-element structure necessary to define the certificate.

1620

1621 **8.4.19 SecurityDetails element**

1622 The ***SecurityDetails*** element defines a set of ***TrustAnchors*** and an associated ***SecurityPolicy*** for
 1623 use in this *CPP*. One or more ***SecurityDetails*** elements can be provided for use in the various
 1624 security functions in the *CPP*. An example of the ***SecurityDetails*** element is:

```
1625
1626     <tp:SecurityDetails tp:securityId="CompanyA_MessageSecurity">
1627         <tp:TrustAnchors tp:trustId="MessageTrustAnchors">
1628             <tp:AnchorCertificateRef tp:certId="TrustedRootCertA3" />
1629             <tp:AnchorCertificateRef tp:certId="TrustedRootCertA5" />
1630         </tp:TrustAnchors>
1631         <tp:SecurityPolicy> ... </tp:SecurityPolicy>
1632     </tp:SecurityDetails>
1633
```

1633

1634 The ***SecurityDetails*** element has zero or one ***TrustAnchors*** element that identifies a set of
 1635 certificates that are trusted by the *Party*. It also has zero or one ***SecurityPolicy*** element.

1636

1637 The ***SecurityDetails*** element allows agreement to be reached on what root certificates will be
 1638 used in checking the validity of the other *Party*'s certificates. It can also specify policy regarding
 1639 operation of the public key infrastructure.

1640

1641 The ***SecurityDetails*** element has one attribute:

- 1642 • A REQUIRED ***securityId*** attribute.

1643

1644 **8.4.19.1 securityId attribute**

1645 The REQUIRED *securityId* attribute is an [XML] ID that is referred to by an element elsewhere
1646 in the *CPP*. Here is an example of how a *SigningSecurityDetailsRef* would refer to the
1647 *SecurityDetails* element shown in the previous section:

```
1648  
1649     <tp:SigningSecurityDetailsRef  
1650 tp:securityId="CompanyA_MessageSecurity" />  
1651
```

1652 **8.4.20 TrustAnchors element**

1653 The *TrustAnchors* element contains one or more *AnchorCertificateRef* elements, each of which
1654 refers to a *Certificate* element (under *PartyInfo*) that represents a certificate trusted by this
1655 *Party*. These trusted certificates are used in the process of certificate path validation. If a
1656 certificate in question does not “chain” to one of this *Party*’s trust anchors, it is considered
1657 invalid.

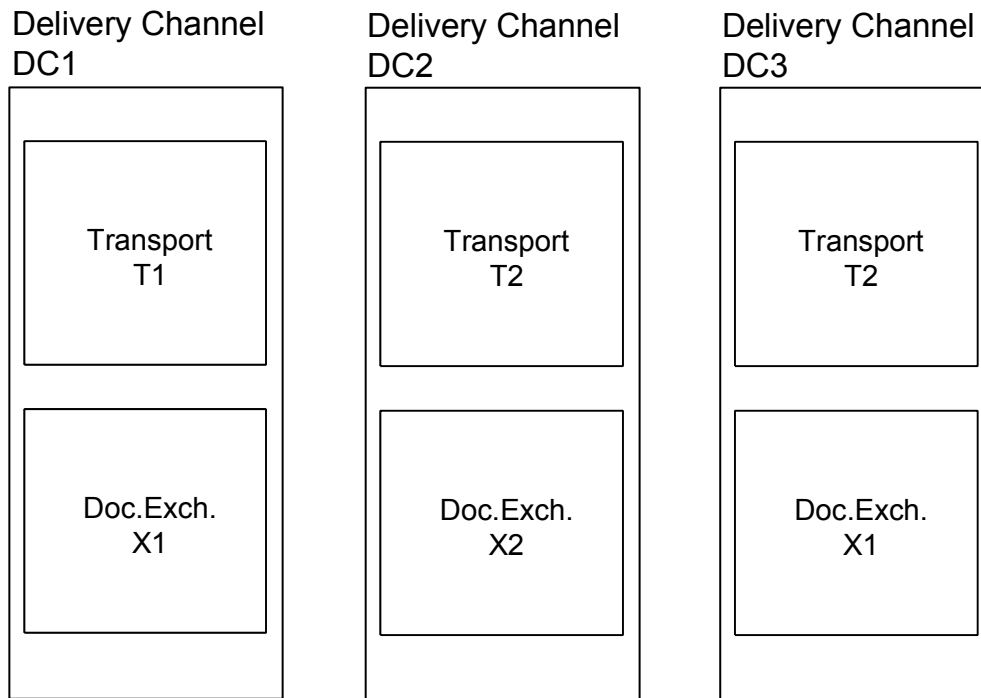
1658
1659 The *TrustAnchors* element eventually resolves into XMLDsig *KeyInfo* elements. These elements
1660 may contain several certificates (a chain), and may refer to those certificates using the
1661 *RetrievalMethod* element. When there is a chain, the trust anchor is the “leaf” certificate with
1662 respect to the “root” issuing certificate authority (CA) certificate. The root CA will be a self-
1663 issued and self-signed certificate, and using the Issuer information and perhaps key usage
1664 attributes, the leaf certificate (“issued but not issuing” within the chain) can be determined. The
1665 chain is included for convenience in that validity checks typically will chain to a “root” CA.
1666 Please note that the inclusion of a root CA in a chain does not mean that the root CA is being
1667 announced as a trust anchor. It is possible for there to be a PKI policy in which some, but not all,
1668 intermediate CAs are trusted. If a root CA were accepted as a trust anchor, all of its intermediate
1669 CAs, and all the certificates they issue, would be validated. That might not be what was intended.
1670

1671 **8.4.21 SecurityPolicy element**

1672 The *SecurityPolicy* element is a placeholder for future apparatus that will enable the *Party* to
1673 specify its policy and compliance regarding specific components of its public key infrastructure.
1674 For example, it might stipulate revocation checking procedures or constraints related to name,
1675 usage, or path length.
1676

1677 **8.4.22 DeliveryChannel element**

1678 A delivery channel is a combination of a *Transport* element and a *DocExchange* element that
1679 describes the *Party's Message* communication characteristics. The *CPP* SHALL contain one or
1680 more *DeliveryChannel* elements, one or more *Transport* elements, and one or more
1681 *DocExchange* elements. Each delivery channel SHALL refer to any combination of a
1682 *DocExchange* element and a *Transport* element. The same *DocExchange* element or the same
1683 *Transport* element can be referred to by more than one delivery channel. Two delivery channels
1684 can use the same transport protocol and the same document-exchange protocol and differ only in
1685 details such as communication addresses or security definitions. Figure 5 illustrates three
1686 delivery channels.

Figure 5: Three Delivery Channels

1687
 1688 The delivery channels have ID attributes with values "DC1", "DC2", and "DC3". Each delivery
 1689 channel contains one transport definition and one document-exchange definition. Each transport
 1690 definition and each document-exchange definition also has an ID attribute whose value is shown
 1691 in the figure. Note that delivery channel DC3 illustrates that a delivery channel can refer to the
 1692 same transport definition and document-exchange definition used by other delivery channels but
 1693 a different combination. In this case delivery channel DC3 is a combination of transport
 1694 definition T2 (also referred to by delivery channel DC2) and document-exchange definition X1
 1695 (also referred to by delivery channel DC1).

1696
 1697 Following is the delivery-channel syntax.

```

1698
1699 <tp:DeliveryChannel
1700     tp:channelId="channel1"
1701     tp:transportId="transport1"
1702     tp:docExchangeId="docExchange1"
1703     <tp:MessagingCharacteristics
1704         tp:syncReplyMode="none"
1705         tp:ackRequested="always"
1706         tp:ackSignatureRequested="always"
1707         tp:duplicateElimination="always"
1708         tp:actor="urn:oasis:names:tc:ebxml-msg:actor:nextMSH"/>
1709 </tp:DeliveryChannel>
1710
  
```

1711 Each *DeliveryChannel* element identifies one *Transport* element and one *DocExchange* element
 1712 that together make up a single delivery channel definition.

1713

1714 The *DeliveryChannel* element has the following attributes:

- 1715 • a REQUIRED *channelId* attribute,
- 1716 • a REQUIRED *transportId* attribute,
- 1717 • a REQUIRED *docExchangeId* attribute.

1718

1719 The *DeliveryChannel* element has one REQUIRED child element, *MessagingCharacteristics*.

1720 •

1721 8.4.22.1 channelId attribute

1722 The *channelId* attribute is an [XML] ID attribute that uniquely identifies the *DeliveryChannel*
1723 element for reference, using IDREF attributes, from other parts of the *CPP* or *CPA*.

1724

1725 8.4.22.2 transportId attribute

1726 The *transportId* attribute is an [XML] IDREF that identifies the *Transport* element that defines
1727 the transport characteristics of the delivery channel. It MUST have a value that is equal to the
1728 value of a *transportId* attribute of a *Transport* element elsewhere within the *CPP* document.

1729

1730 8.4.22.3 docExchangeId attribute

1731 The *docExchangeId* attribute is an [XML] IDREF that identifies the *DocExchange* element that
1732 defines the document-exchange characteristics of the delivery channel. It MUST have a value
1733 that is equal to the value of a *docExchangeId* attribute of a *DocExchange* element elsewhere
1734 within the *CPP* document.

1735

1736 8.4.23 MessagingCharacteristics element

1737 The *MessagingCharacteristics* element describes the attributes associated with messages
1738 delivered over a given delivery channel. The collaborating *Parties* can stipulate that these
1739 attributes be fixed for all messages sent through the delivery channel, or they can agree that these
1740 attributes be variable on a “per message” basis.

1741

1742 *CPP* and *CPA* composition tools and *CPA* deployment tools SHALL check the delivery channel
1743 definition (transport and document-exchange) for consistency with these attributes.

1744

1745 The *MessagingCharacteristics* element has the following attributes:

- 1746 • An IMPLIED *syncReplyMode* attribute,
- 1747 • an IMPLIED *ackRequested* attribute,
- 1748 • an IMPLIED *ackSignatureRequested* attribute,
- 1749 • an IMPLIED *duplicateElimination* attribute,
- 1750 • an IMPLIED *actor* attribute.

1751

1752 8.4.23.1 syncReplyMode attribute

1753 The *syncReplyMode* attribute is an enumeration comprised of the following possible values:

- 1754 • "mshSignalsOnly"
- 1755 • "signalsOnly"
- 1756 • "responseOnly"

- 1757 • "signalsAndResponse"
 1758 • "none"

1759
 1760 This attribute, when present, indicates what the sending application expects in a synchronous
 1761 response (the delivery channel **MUST** be bound to a synchronous communication protocol such
 1762 as HTTP when *syncReplyMode* is not "none").

1763
 1764 The value of "mshSignalsOnly" indicates that the response returned (on the HTTP 200 response
 1765 in the case of HTTP) will only contain standalone *Message Service Handler (MSH)* level
 1766 messages like Acknowledgment (for Reliable Messaging) and Error messages. All other
 1767 application level responses are to be returned asynchronously (using a *DeliveryChannel* element
 1768 determined by the service and action in question).

1769
 1770 The value of "signalsOnly" indicates that the response returned (on the HTTP 200 response in
 1771 the case of HTTP) will only include one or more *Business* signals as defined in the *Process-*
 1772 *Specification* document[ebBPSS], plus any piggybacked MSH level signals, but not a *Business-*
 1773 *response Message*. If the *Process-Specification* calls for the use of a *Business-response Message*,
 1774 then the latter **MUST** be returned asynchronously. If the *Business Process* does not call for the
 1775 use of an *Acceptance Acknowledgment* signal, then the *Action* element in the synchronously
 1776 returned ebXML *Message* **MUST** be set to "ReceiptAcknowledgment". Otherwise, the *Action*
 1777 element in the synchronously returned ebXML *Message* (which includes both a *Receipt*
 1778 *Acknowledgment* signal and an *Acceptance Acknowledgment* signal) **MUST** be set to
 1779 "AcceptanceAcknowledgment".

1780
 1781 The value of "responseOnly" indicates that any *Business* signals, even if they are indicated in the
 1782 *Process Specification*, are to be omitted and only the *Business-response Message* will be returned
 1783 synchronously, plus any piggybacked MSH level signals. To be consistent, the
 1784 *timeToAcknowledgeReceipt* and *timeToAcknowledgeAcceptance* attributes under the
 1785 corresponding *BusinessTransactionCharacteristics* element **SHOULD** be set to zero to indicate
 1786 that these signals are not to be used at all. The *Action* element in the synchronously returned
 1787 ebXML *Message* is determined by the name of the action in the *CPA* that corresponds to the
 1788 appropriate *RespondingBusinessActivity* in the *Business Process*.

1789
 1790 The value of "signalsAndResponse" indicates that the application will synchronously return the
 1791 *Business-response Message* in addition to one or more *Business* signals, plus any piggybacked
 1792 *MSH* level signals. In this case, each signal and response that is bundled into the same ebXML
 1793 message must appear as a separate MIME part (i.e., be placed in a separate payload container).
 1794 To be consistent, the *timeToAcknowledgeReceipt* and *timeToPerform* attributes under the
 1795 corresponding *BusinessTransactionCharacteristics* element **SHOULD** have identical values.
 1796 The *timeToAcknowledgeAcceptance* attribute, if specified, **SHOULD** also have the same value
 1797 as the above two timing attributes. The *Action* element in the synchronously returned ebXML
 1798 *Message* is determined by the name of the action in the *CPA* that corresponds to the appropriate
 1799 *RespondingBusinessActivity* in the *Business Process*.

1800
 1801 The *Receipt Acknowledgment* signal for the *Business-response Message*, sent from the request

1802 initiator back to the responder, if called for by the *Process-Specification*, MUST also be
1803 delivered over the same synchronous connection.

1804
1805 NOTE: For HTTP 1.1 clients and servers, two HTTP requests and replies will have to be
1806 sent and received on the same connection. Implementations that implicitly assume that a
1807 HTTP connection will be closed after a single synchronous request reply interchange will
1808 not be able to support the "signalsAndResponse" synchronous reply mode.

1809
1810 The value of "none", which is the implied default value in the absence of the *syncReplyMode*
1811 attribute, indicates that neither the *Business-response Message* nor any *Business* signal(s) will be
1812 returned synchronously. In this case, all *Message Service Handler* level and *Business* level
1813 messages will be returned as separate asynchronous messages.

1814
1815 The ebXML *Message Service's SyncReply* element is included in the SOAP Header whenever
1816 the *syncReplyMode* attribute has a value other than "none". If the delivery channel identifies a
1817 transport protocol that has no synchronous capabilities (such as SMTP), the
1818 *BusinessTransactionCharacteristics* element SHALL NOT have a *syncReplyMode* attribute
1819 with a value other than "none".

1820
1821 When the value of the *syncReplyMode* attribute is other than "none", a synchronous delivery
1822 channel SHALL be used to exchange all messages necessary for conducting a business
1823 transaction. If the *Process Specification* calls for the use of non-repudiation of receipt for the
1824 response message, then the initiator is expected to return a signed *ReceiptAcknowledgment* signal
1825 for the responder's response message.

1826
1827 **8.4.23.2 ackRequested attribute**
1828 The IMPLIED *ackRequested* attribute is an enumeration comprised of the following possible
1829 values:

- 1830 • "always"
- 1831 • "never"
- 1832 • "perMessage"

1833
1834 This attribute has the default value "perMessage" meaning whether the *AckRequested* element in
1835 the SOAP Header is present or absent can be varied on a "per message" basis. If this attribute is
1836 set to "always", then every message sent over the delivery channel MUST have an *AckRequested*
1837 element in the SOAP Header. If this attribute is set to "never", then every message sent over the
1838 delivery channel MUST NOT have an *AckRequested* element in the SOAP Header.

1839
1840 If the *ackRequested* attribute is not set to "never", then the *ReliableMessaging* element must be
1841 present under the corresponding *DocExchange* element to provide the necessary Reliable
1842 Messaging parameters.

1843
1844 **8.4.23.3 ackSignatureRequested attribute**
1845 The IMPLIED *ackSignatureRequested* attribute is an enumeration comprised of the following
1846 possible values:

- 1847 • "always"
- 1848 • "never"
- 1849 • "perMessage"

1850

1851 This attribute determines how the *signed* attribute within the *AckRequested* element in the SOAP
1852 Header is to be set. It has the default value "perMessage" meaning that the *signed* attribute in the
1853 *AckRequested* element within the SOAP Header can be set to "true" or "false" on a "per
1854 message" basis. If this attribute is set to "always", then every message sent over the delivery
1855 channel that has an *AckRequested* element in the SOAP Header MUST have its *signed* attribute
1856 set to "true". If this attribute is set to "never", then every message sent over the delivery channel
1857 that has an *AckRequested* element in the SOAP Header MUST have its *signed* attribute set to
1858 "false". If the *ackRequested* attribute is set to "never", the setting of the *ackSignatureRequested*
1859 attribute has no effect.

1860

1861 NOTE: By enabling the use of signed *Acknowledgment* for reliably delivered messages,
1862 a weak form of non-repudiation of receipt can be supported. This is considered weaker
1863 than the *Receipt Acknowledgment* signal because no schema check can be performed on
1864 the payload prior to the return of the *Acknowledgment*. The *ackSignatureRequested*
1865 attribute can be set independent of the value for the *isNonRepudiationReceiptRequired*
1866 attribute under the *BusinessTransactionCharacteristics* element. Thus, even if the
1867 original *Process-Specification* specifies that non-repudiation of receipt is to be
1868 performed, the *CPP* and/or *CPA* can override this requirement, set
1869 *isNonRepudiationReceiptRequired* to "false" and *ackSignatureRequested* to "always"
1870 and thereby achieve the weak form of non-repudiation of receipt.

1871

1872 8.4.23.4 duplicateElimination attribute

1873 The IMPLIED *duplicateElimination* attribute is an enumeration comprised of the following
1874 possible values:

- 1875 • "always"
- 1876 • "never"
- 1877 • "perMessage"

1878

1879 This attribute determines whether the *DuplicateElimination* element within the *MessageHeader*
1880 element in the SOAP Header is to be present. It has the default value "perMessage" meaning that
1881 the *DuplicateElimination* element within the SOAP Header can be present or absent on a "per
1882 message" basis. If this attribute is set to "always", then every message sent over the delivery
1883 channel MUST have a *DuplicateElimination* element in the SOAP Header. If this attribute is set
1884 to "never", then every message sent over the delivery channel MUST NOT have a
1885 *DuplicateElimination* element in the SOAP Header. If the *duplicateElimination* attribute is not
1886 set to "never", then the *PersistDuration* element must be present under the corresponding
1887 *DocExchange* element to provide the necessary persistent storage parameter.

1888

1889 8.4.23.5 actor attribute

1890 The IMPLIED *actor* attribute is an enumeration of the following possible values:

- 1891 • "urn:oasis:names:tc:ebxml-msg:actor:nextMSH"

- "urn:oasis:names:tc:ebxml-msg:actor:toPartyMSH"

This is a URI that will be used as the value for the *actor* attribute in the *AckRequested* element (see [ebMS]) in case the latter is present in the SOAP Header, as governed by the *ackRequested* attribute within the *MessagingCharacteristics* element in the *CPA*. If the *ackRequested* attribute is set to "never", the setting of the *actor* attribute has no effect.

8.4.24 Transport element

The *Transport* element defines the *Party's* network communication capabilities. One or more *Transport* elements MUST be present in a *CPP*, each of which describes a mechanism the *Party* uses to send messages, a mechanism it uses to receive messages, or both. The following example illustrates the structure of a typical *Transport* element:

```

1904
1905     <tp:Transport tp:transportId="transportA1">
1906         <tp:TransportSender <!-- 0 or 1 time -->
1907             <tp:TransportProtocol tp:version="1.1">HTTP</tp:Protocol>
1908             <tp:TransportClientSecurity>
1909                 <tp:TransportSecurityProtocol tp:version="3.0">
1910                     SSL
1911                 </tp:TransportSecurityProtocol>
1912                 <tp:ClientCertificateRef tp:certId="CompanyA_ClientCert"/>
1913                 <tp:ServerSecurityDetailsRef
1914                     tp:securityId="CompanyA_TransportSecurity"/>
1915             </tp:TransportClientSecurity>
1916         </tp:TransportSender>
1917         <tp:TransportReceiver <!-- 0 or 1 time -->
1918             <tp:TransportProtocol tp:version="1.1">HTTP</tp:Protocol>
1919             <tp:Endpoint
1920                 tp:uri="https://www.CompanyA.com/servlets/ebxmlhandler"
1921                 tp:type="allPurpose"/>
1922             <tp:TransportServerSecurity>
1923                 <tp:TransportSecurityProtocol tp:version="3.0">
1924                     SSL
1925                 </tp:TransportSecurityProtocol>
1926                 <tp:ServerCertificateRef tp:certId="CompanyA_ServerCert"/>
1927                 <tp:ClientSecurityDetailsRef
1928                     tp:securityId="CompanyA_TransportSecurity"/>
1929             </tp:TransportServerSecurity>
1930         </tp:TransportReceiver>
1931     </tp:Transport>
1932

```

The *Transport* element consists of zero or one *TransportSender* element and zero or one *TransportReceiver* element.

A *Transport* that contains both *TransportSender* and *TransportReceiver* elements is said to be *bi-directional* in that it can be used for send and receiving messages. If the *Party* prefers to communicate in synchronous mode (where replies are returned over the same TCP connections messages are sent on; see Section 8.4.23.1), its *CPP* MUST provide a *ServiceBinding* that contains *ActionBindings* that are bound to a *DeliveryChannel* that uses a bi-directional *Transport*.

1944 A **Transport** that contains either a **TransportSender** or a **TransportReceiver** element, but not
 1945 both, is said to be *unidirectional*. A unidirectional **Transport** can only be used for sending or
 1946 receiving messages (not both) depending on which element it includes.

1947
 1948 A *CPP* contains as many **Transport** elements as are needed to fully express the *Party*'s inbound
 1949 and outbound communication capabilities. If, for example, the *Party* can send and receive
 1950 messages via HTTP and SMTP, its *CPP* would contain a **Transport** element containing its HTTP
 1951 properties and another **Transport** element containing its SMTP properties.

1952
 1953 The **Transport** element has

- 1954 • a REQUIRED *transportId* attribute.

1955 1956 **8.4.24.1 transportId attribute**

1957 The REQUIRED *transportId* attribute is an [XML] ID that refers to a **Transport** element
 1958 elsewhere in the *CPP*. Here is an example of a **DeliveryChannel** that refers to the **Transport**
 1959 element shown in the previous section:

```
1960
1961 <tp:DeliveryChannel tp:channelId="channelA1"
1962     tp:transportId="transportA1"
1963     tp:docExchangeId="docExchangeA1">
1964     <tp:MessagingCharacteristics . . . />
1965 </tp:DeliveryChannel>
1966
```

1967 **8.4.25 TransportSender element**

1968 The **TransportSender** element contains properties related to the sending side of a
 1969 **DeliveryChannel**. Its REQUIRED **TransportProtocol** element specifies the transport protocol
 1970 that will be used for sending messages. The **AccessAuthentication** element(s), if present,
 1971 specifies the type(s) of access authentication supported by the client. The
 1972 **TransportClientSecurity** element, if present, defines the *Party*'s provisions for client-side
 1973 transport layer security.

1974
 1975 The **TransportSender** element has no attributes.

1976 1977 **8.4.26 TransportProtocol element**

1978 The **TransportProtocol** element identifies a transport protocol that the *Party* is capable of using
 1979 to send or receive *Business* data. The IMPLIED *version* attribute identifies the specific version
 1980 of the protocol.

1981
 1982 NOTE: It is the aim of this specification to enable support for any transport capable of
 1983 carrying MIME content using the vocabulary defined herein.

1984 1985 **8.4.27 AccessAuthentication element**

1986 The **AccessAuthentication** element, if present, indicates the authentication mechanism that MAY
 1987 be used by a transport server to challenge a client request and by a client to provide

1988 authentication information to a server. For example, [RFC2617] specifies two access
 1989 authentication schemes for HTTP: "basic" and "digest". A client that supports both would have
 1990 two *AccessAuthentication* elements, as shown below. When multiple schemes are supported, the
 1991 order in which they are specified in the CPP indicates the order of preference.

```

1992     <tp:TransportSender>
1993       <tp:TransportProtocol tp:version="1.1">HTTP</tp:TransportProtocol>
1994       <tp:AccessAuthentication>digest</tp:AccessAuthentication>
1995       <tp:AccessAuthentication>basic</tp:AccessAuthentication>
1996       <tp:TransportClientSecurity>
1997         ...
1998       </tp:TransportClientSecurity>
1999     </tp:TransportSender>
2000
2001
  
```

2002 NOTE: A *CPA* will contain, for each *TransportSender* or *TransportReceiver*, only the
 2003 agreed-upon *AccessAuthentication* elements.

2004
 2005 NOTE: For basic authentication, the userid and password values are configured through
 2006 means outside of this specification.

2008 8.4.28 TransportClientSecurity element

2009 The *TransportClientSecurity* element provides information about this *Party*'s transport client
 2010 needed by the other *Party*'s transport server to enable a secure connection to be established
 2011 between the two. It contains a REQUIRED *TransportSecurityProtocol* element, zero or one
 2012 *ClientCertificateRef* element, zero or one *ServerSecurityDetailsRef* element, and zero or more
 2013 *EncryptionAlgorithm* elements.

2014
 2015 In asynchronous messaging mode, the sender will always be a client to the receiver's server. In
 2016 synchronous messaging mode, the MSH-level reply (and maybe a bundled business signal and/or
 2017 business response) is sent back over the same connection the initial business message arrived on.
 2018 In such cases, where the sender is the server and the receiver is the client and the connection
 2019 already exists, the sender's *TransportClientSecurity* and the receiver's *TransportServerSecurity*
 2020 elements SHALL be ignored.

2022 8.4.29 TransportSecurityProtocol element

2023 The *TransportSecurityProtocol* element identifies the transport layer security protocol that is
 2024 supported by the parent *Transport*. The IMPLIED *version* attribute identifies the specific version
 2025 of the protocol.

2026
 2027 For encryption, the protocol is TLS Version 1.0[RFC2246], which uses public-key encryption.
 2028 Appendix E of the TLS Version 1.0 specification[RFC2246] covers backward compatibility with
 2029 SSL [SSL].

2031 8.4.30 ClientCertificateRef element

2032 The *ClientCertificateRef* element identifies the certificate to be used by the client's transport
 2033 security module. The REQUIRED IDREF attribute *certId* identifies the certificate to be used by

2034 referring to the *Certificate* element (under *PartyInfo*) that has the matching ID attribute value. A
2035 TLS-capable HTTP client, for example, uses this certificate to authenticate itself with receiver's
2036 secure HTTP server.

2037
2038 The *ClientCertificateRef* element, if present, indicates that mutual authentication between client
2039 and server (i.e., initiator and responder of the HTTP connection) MUST be performed.

2040 The *ClientCertificateRef* element has

- 2041 • A REQUIRED *certId* attribute.
- 2042 •
- 2043 •

2044 **8.4.31 ServerSecurityDetailsRef element**

2045 The *ServerSecurityDetailsRef* element identifies the trust anchors and security policy that this
2046 *Party* will apply to the other *Party*'s server authentication certificate.

2047
2048 The *ServerSecurityDetailsRef* element has

- 2049 • A REQUIRED *securityId* attribute.
- 2050 •

2051 **8.4.32 Encryption Algorithm**

2052 Zero or more *EncryptionAlgorithm* elements may be included under the
2053 *TransportClientSecurity* or *TransportServerSecurity* element. Multiple elements are of more
2054 use in a CPP context, to announce capabilities or preferences; normally, a CPA will contain the
2055 agreed upon context. When zero or more than one element is present in a CPA, the *Parties* agree
2056 to allow the automatic negotiation capability of the *TransportSecurityProtocol* element to
2057 determine the actual algorithm used.

2058
2059 The elements' ordering will reflect the preference for algorithms. A primary reason for including
2060 this element is to permit use of the *minimumStrength* attribute; a large value for this attribute
2061 can indicate that high encryption strength is desired or has been agreed upon for the
2062 *TransportSecurityProtocol*.

2063
2064 See section 8.4.50 for the full description of this element.

2065
2066 For SSL and TLS, it is customary to specify cipher suite values as text values for the
2067 *EncryptionAlgorithm* element. These values include, but are not limited to:

- 2068 •
- 2069 • SSL_RSA_FIPS_WITH_3DES_EDE_CBC_SHA,
- 2070 • TLS_RSA_WITH_3DES_EDE_CBC_SHA,
- 2071 • SSL_RSA_WITH_3DES_EDE_CBC_SHA,
- 2072 • SSL_RSA_WITH_RC4_128_MD5,
- 2073 • SSL_RSA_WITH_RC4_128_SHA,
- 2074 • SSL_DH_DSS_WITH_3DES_EDE_CBC_SHA,
- 2075 • SSL_DHE_RSA_WITH_3DES_EDE_CBC_SHA.
- 2076 •

2077 Consult the original specifications for enumerations and discussions of these values.
2078

2079 **8.4.33 TransportReceiver element**

2080 The *TransportReceiver* element contains properties related to the receiving side of a
2081 *DeliveryChannel*. Its REQUIRED *TransportProtocol* element specifies the transport protocol
2082 that will be used for receiving messages. One or more REQUIRED *Endpoint* elements specify
2083 logical addresses where messages can be received. The *AccessAuthentication* element(s), if
2084 present, indicates the type(s) of access authentication supported by the server. Zero or one
2085 *TransportServerSecurity* element defines the *Party*'s provisions for server-side transport layer
2086 security.

2087
2088 The *TransportReceiver* element has no attributes.
2089

2090 **8.4.34 Endpoint element**

2091 One or more *Endpoint* elements SHALL be provided for each *TransportReceiver* element. Each
2092 *Endpoint* specifies a logical address and an indication of what kinds of messages can be received
2093 at that location.

2094
2095 Each *Endpoint* has the following attributes:

- 2096 • a REQUIRED *uri* attribute,
- 2097 • an IMPLIED *type* attribute.
- 2098 •

2099 **8.4.34.1 uri attribute**

2100 The REQUIRED *uri* attribute specifies a URI identifying the address of a resource. The value of
2101 the *uri* attribute SHALL conform to the syntax for expressing URIs as defined in [RFC2396].
2102

2103 **8.4.34.2 type attribute**

2104 The *type* attribute identifies the purpose of this endpoint. The value of *type* is an enumeration;
2105 permissible values are "login", "request", "response", "error", and "allPurpose". There can be, at
2106 most, one of each. If the *type* attribute is omitted, its value defaults to "allPurpose". The "login"
2107 endpoint is used for the address for the initial *Message* between the two *Parties*. The "request"
2108 and "response" endpoints are used for request and response *Messages*, respectively. To enable
2109 error *Messages* to be received, each *Transport* element SHALL contain at least one endpoint of
2110 type "error", "response", or "allPurpose".
2111

2112 The types of *Endpoint* element within a *TransportReceiver* element MUST not be overlapping.
2113 Thus, it would be erroneous to include both an "allPurpose" *Endpoint* element along with
2114 another *Endpoint* element of any type.
2115

2116 **8.4.35 TransportServerSecurity element**

2117 The *TransportServerSecurity* element provides information about this *Party*'s transport server
2118 needed by the other *Party*'s transport client to enable a secure connection to be established

2119 between the two. It contains a REQUIRED *TransportSecurityProtocol* element, a REQUIRED
 2120 *ServerCertificateRef* element, zero or one *ClientSecurityDetailsRef* element, and zero or more
 2121 *EncryptionAlgorithm* elements. See Section 8.4.32 for a description of the
 2122 *EncryptionAlgorithm* element.

2123

2124 NOTE: See the note in Section 8.4.27 regarding the relevance of the
 2125 *TransportServerSecurity* element when synchronous replies are in use.

2126

2127 **8.4.36 ServerCertificateRef element**

2128 The *ServerCertificateRef* element, if present, identifies the certificate to be used by the server's
 2129 transport security module. The REQUIRED IDREF attribute *certId* identifies the certificate to be
 2130 used by referring to the *Certificate* element (under *PartyInfo*) that has the matching ID attribute
 2131 value. A TLS-enabled HTTP server, for example, uses this certificate to authenticate itself with
 2132 the sender's TLS client.

2133

2134 The *ServerCertificateRef* element MUST be present if the transport security protocol uses
 2135 certificates. It MAY be omitted otherwise (e.g. if authentication is by password).

2136

2137 The *ServerCertificateRef* element has

- 2138 • A REQUIRED *certId* attribute.

2139

2140 **8.4.37 ClientSecurityDetailsRef element**

2141 The *ClientSecurityDetailsRef* element, if present, identifies the trust anchors and security policy
 2142 that this *Party* will apply to the other *Party*'s client authentication certificate.

2143

2144 The *ClientSecurityDetailsRef* element has

- 2145 • A REQUIRED *securityId* attribute.

2146

2147 **8.4.38 Transport protocols**

2148 In the following sections, we discuss the specific details of each supported transport protocol.

2149

2150 **8.4.38.1 HTTP**

2151 HTTP is Hypertext Transfer Protocol[HTTP]. For HTTP, the endpoint is a URI that SHALL
 2152 conform to [RFC2396]. Depending on the application, there MAY be one or more endpoints,
 2153 whose use is determined by the application.

2154

2155 Following is an example of an HTTP endpoint:

2156

```
2157 <tp:Endpoint tp:uri="http://example.com/servlet/ebxmlhandler"  
2158 tp:type="request"/>
```

2159

2160 The "request" and "response" endpoints can be dynamically overridden for a particular request
2161 or asynchronous response by application-specified URIs in *Business* documents exchanged under
2162 the *CPA*.

2163
2164 For a synchronous response, the "response" endpoint is ignored if present. A synchronous
2165 response is always returned on the existing connection, i.e. to the URI that is identified as the
2166 source of the connection.

2167 2168 **8.4.38.2 SMTP**

2169 SMTP is Simple Mail Transfer Protocol[SMTP]. For use with this standard, Multipurpose
2170 Internet Mail Extensions[MIME] MUST be supported. For SMTP, the communication address is
2171 the fully qualified mail address of the destination *Party* as defined by [RFC2822]. Following is
2172 an example of an SMTP endpoint:

```
2173 <tp:Endpoint tp:uri="mailto:ebxmlhandler@example.com"  
2174 tp:type="request" />
```

2176
2177 NOTE: The SMTP Mail Transfer Agent (MTA) can encode binary data when the
2178 receiving MTA does not support binary transfer. In general, SMTP transfer may involve
2179 coding and recoding of Content-Transfer-Encodings as a message moves along a
2180 sequence of MTAs. Such changes can in some circumstances invalidate some kinds of
2181 signatures even though no malicious actions or transmission errors have occurred.

2182
2183 NOTE: SMTP by itself (without any authentication or encryption) is subject to denial of
2184 service and masquerading by unknown *Parties*. It is strongly suggested that those *Parties*
2185 who choose SMTP as their transport layer also choose a suitable means of encryption and
2186 authentication either in the document-exchange layer or in the transport layer such as
2187 [S/MIME].

2188
2189 NOTE: SMTP is an asynchronous protocol that does not guarantee a particular quality of
2190 service. A transport-layer acknowledgment (i.e. an SMTP acknowledgment) to the
2191 receipt of a mail *Message* constitutes an assertion on the part of the SMTP server that it
2192 knows how to deliver the mail *Message* and will attempt to do so at some point in the
2193 future. However, the *Message* is not hardened and might never be delivered to the
2194 recipient. Furthermore, the sender will see a transport-layer acknowledgment only from
2195 the nearest node. If the *Message* passes through intermediate nodes, SMTP does not
2196 provide an end-to-end acknowledgment. Therefore receipt of an SMTP
2197 acknowledgement does not guarantee that the *Message* will be delivered to the
2198 application and failure to receive an SMTP acknowledgment is not evidence that the
2199 *Message* was not delivered. It is RECOMMENDED that the reliable-messaging protocol
2200 in the ebXML *Message* Service be used with SMTP.

2201 2202 **8.4.38.3 FTP**

2203 FTP is File Transfer Protocol[RFC959].

2204
2205 Each *Party* sends a *Message* using FTP PUT. The endpoint specifies the user id and input

2206 directory path (for PUTs to this *Party*). An example of an FTP endpoint is:

```
2207  
2208     <tp:Endpoint uri="ftp://userid@server.foo.com"  
2209         tp:type="request"/>
```

2210
2211 Since FTP needs to be compatible across all implementations, the FTP for ebXML will use the
2212 minimum sets of commands and parameters available for FTP as specified in [RFC959], Section
2213 5.1, and modified in [RFC1123], Section 4.1.2.13. The mode SHALL be stream only and the
2214 type MUST be ASCII Non-print (AN), Image (I) (binary), or Local 8 (L 8) (binary between 8-bit
2215 machines and machines with 36 bit words – for an 8-bit machine Local 8 is the same as Image).

2216
2217 Stream mode closes the data connection upon end of file. The server side FTP MUST set control
2218 to "PASV" before each transfer command to obtain a unique port pair if there are multiple third
2219 party sessions.

2220
2221 NOTE: [RFC 959] states that User-FTP SHOULD send a PORT command to assign a
2222 non-default data port before each transfer command is issued to allow multiple transfers
2223 during a single FTP because of the long delay after a TCP connection is closed until its
2224 socket pair can be reused.

2225
2226 NOTE: The format of the 227 reply to a PASV command is not well standardized and an
2227 FTP client might assume that the parentheses indicated in [RFC959] will be present when
2228 in some cases they are not. If the User-FTP program doesn't scan the reply for the first
2229 digit of host and port numbers, the result will be that the User-FTP might point at the
2230 wrong host. In the response, the h1, h2, h3, h4 is the IP address of the server host and the
2231 p1, p2 is a non-default data transfer port that PASV has assigned.

2232
2233 NOTE: As a recommendation for firewall transparency, [RFC1579] proposes that the
2234 client sends a PASV command, allowing the server to do a passive TCP open on some
2235 random port, and inform the client of the port number. The client can then do an active
2236 open to establish the connection.

2237
2238 NOTE: Since STREAM mode closes the data connection upon end of file, the receiving
2239 FTP might assume abnormal disconnect if a 226 or 250 control code hasn't been received
2240 from the sending machine.

2241
2242 NOTE: [RFC1579] also makes the observation that it might be worthwhile to enhance the
2243 FTP protocol to have the client send a new command APSV (all passive) at startup that
2244 would allow a server that implements this option to always perform a passive open. A
2245 new reply code 151 would be issued in response to all file transfer requests not preceded
2246 by a PORT or PASV command; this *Message* would contain the port number to use for
2247 that transfer. A PORT command could still be sent to a server that had previously
2248 received APSV; that would override the default behavior for the next transfer operation,
2249 thus permitting third-party transfers.

2250

2251 **8.4.39 DocExchange Element**

2252 The **DocExchange** element provides information that the *Parties* MUST agree on regarding
 2253 exchange of documents between them. This information includes the messaging service
 2254 properties (e.g. ebXML *Message Service*[ebMS]).

2255
 2256 Following is the structure of the **DocExchange** element of the *CPP*. Subsequent sections
 2257 describe each child element in greater detail.

```

2258
2259 <tp:DocExchange tp:docExchangeId="docExchangeB1">
2260   <tp:ebXMLSenderBinding tp:version="2.0"> <!-- 0 or 1 -->
2261     <tp:ReliableMessaging> <!-- 0 or 1 -->
2262       . . .
2263     </tp:ReliableMessaging>
2264     <tp:PersistDuration> <!-- 0 or 1 -->
2265       . . .
2266     </tp:PersistDuration>
2267     <tp:SenderNonRepudiation> <!-- 0 or 1 -->
2268       . . .
2269     </tp:SenderNonRepudiation>
2270     <tp:SenderDigitalEnvelope> <!-- 0 or 1 -->
2271       . . .
2272     </tp:SenderDigitalEnvelope>
2273     <tp:NamespaceSupported> <!-- 0 or more -->
2274       . . .
2275     </tp:NamespaceSupported>
2276   </tp:ebXMLSenderBinding>
2277   <tp:ebXMLReceiverBinding tp:version="2.0"> <!-- 0 or 1 -->
2278     <tp:ReliableMessaging> <!-- 0 or 1 -->
2279       . . .
2280     </tp:ReliableMessaging>
2281     <tp:PersistDuration> <!-- 0 or 1 -->
2282       . . .
2283     </tp:PersistDuration>
2284     <tp:ReceiverNonRepudiation> <!-- 0 or 1 -->
2285       . . .
2286     </tp:ReceiverNonRepudiation>
2287     <tp:ReceiverDigitalEnvelope> <!-- 0 or 1 -->
2288       . . .
2289     </tp:ReceiverDigitalEnvelope>
2290     <tp:NamespaceSupported> <!-- 0 or more -->
2291       . . .
2292     </tp:NamespaceSupported>
2293   </tp:ebXMLReceiverBinding>
2294 </tp:DocExchange>
2295

```

2296 The **DocExchange** element is comprised of zero or one **ebXMLSenderBinding** child element
 2297 and zero or one **ebXMLReceiverBinding** child element. It MUST have at least one child
 2298 element. *CPP* and *CPA* composition tools and *CPA* deployment tools SHALL verify the
 2299 presence of a child element.

2300
 2301 NOTE: The document-exchange section can be extended to messaging services other
 2302 than the ebXML *Message* service by adding additional **xxxSenderBinding** and
 2303 **xxxReceiverBinding** elements and their child elements that describe the other services,
 2304 where *xxx* is replaced by the name of the additional binding. An example is
 2305 **XMLPSenderBinding/XMLPReceiverBinding**, which might define support for the future

2306 XML Protocol specification.

2307

2308 **8.4.39.1 docExchangeId attribute**

2309 The *DocExchange* element has a single REQUIRED *docExchangeId* attribute that is an [XML]
2310 ID that provides a unique identifier that can be referenced from elsewhere within the *CPP*
2311 document.

2312

2313 **8.4.40 ebXMLSenderBinding element**

2314 The *ebXMLSenderBinding* element describes properties related to sending messages with the
2315 ebXML *Message Service*[ebMS]. The *ebXMLSenderBinding* element is comprised of the
2316 following child elements:

- 2317 • zero or one *ReliableMessaging* element which specifies the characteristics of reliable
2318 messaging,
- 2319 • zero or one *PersistDuration* element which specifies the duration for which certain
2320 messages have to be stored persistently for the purpose of duplicate elimination,
- 2321 • zero or one *SenderNonRepudiation* element which specifies the sender's requirements
2322 and certificate for message signing,
- 2323 • zero or one *SenderDigitalEnvelope* element which specifies the sender's requirements
2324 for encryption by the digital-envelope[DIGENV] method,
- 2325 • zero or more *NamespaceSupported* elements that identify any namespace extensions
2326 supported by the messaging service implementation.

2327

2328 The *ebXMLSenderBinding* element has one attribute:

- 2329 • a REQUIRED *version* attribute.

2330 NOTE: A CPA could be valid even when omitting all children under

2331 *ebXMLSenderBinding*.

2332

2333 **8.4.40.1 version attribute**

2334 The REQUIRED *version* attribute identifies the version of the ebXML *Message Service*
2335 specification being used.

2336

2337 **8.4.41 ReliableMessaging element**

2338 The *ReliableMessaging* element specifies the properties of reliable ebXML *Message* exchange.
2339 The default that applies if the *ReliableMessaging* element is omitted is "BestEffort". The
2340 following is the element structure:

2341

```
2342 <tp:ReliableMessaging>
2343   <tp:Retries>5</tp:Retries>
2344   <tp:RetryInterval>PT2H</tp:RetryInterval>
2345   <tp:MessageOrderSemantics>Guaranteed</tp:MessageOrderSemantics>
2346 </tp:ReliableMessaging>
```

2347

2348 Semantics of reliable messaging are explained in the ebXML *Message Service*
2349 specification[ebMS] chapter on Reliable Messaging Combinations.

2350

2351 The **ReliableMessaging** element is comprised of the following child elements.

- 2352 • zero or one **Retries** element,
- 2353 • zero or one **RetryInterval** element,
- 2354 • a REQUIRED MessageOrderSemantics element.

2355

2356 8.4.41.1 Retries and RetryInterval elements

2357 The **Retries** and **RetryInterval** elements specify the permitted number of retries and the interval,
2358 expressed as an XML Schema[XMLSCHEMA-2] duration, between retries of sending a reliably
2359 delivered *Message* following a timeout waiting for the *Acknowledgment*. The purpose of the
2360 **RetryInterval** element is to improve the likelihood of success on retry by deferring the retry until
2361 any temporary conditions that caused the error might be corrected. The **RetryInterval** applies to
2362 the time between sending of the original message and the first retry, as well as the time between
2363 all subsequent retries.

2364

2365 The **Retries** and **RetryInterval** elements MUST either be included together or be omitted
2366 together. If they are omitted, the values of the corresponding quantities (number of retries and
2367 retry interval) are a local matter at each *Party*.

2368

2369 8.4.41.2 MessageOrderSemantics element

2370 The **MessageOrderSemantics** element is an enumeration comprised of the following possible
2371 values:

- 2372 • "Guaranteed"
- 2373 • "NotGuaranteed"

2374

2375 The presence of a **MessageOrderSemantics** element in the SOAP Header for ebXML messages
2376 determines if the ordering of messages sent from the *From Party* needs to be preserved so that
2377 the *To Party* receives those messages in the order in which they were sent. If the
2378 **MessageOrderSemantics** element is set to "Guaranteed", then the ebXML message MUST
2379 contain a **MessageOrder** element in the SOAP Header. If the **MessageOrderSemantics** element
2380 is set to "NotGuaranteed", then the ebXML message MUST NOT contain a **MessageOrder**
2381 element in the SOAP Header. Guaranteed message ordering implies the use of duplicate
2382 elimination. Therefore, the **PersistDuration** element MUST also appear if
2383 **MessageOrderSemantics** is set to "Guaranteed".

2384

2385 8.4.42 PersistDuration element

2386 The value of the **PersistDuration** element is the minimum length of time, expressed as an XML
2387 Schema[XMLSCHEMA-2] duration, that data from a *Message* that is sent reliably is kept in
2388 *Persistent Storage* by an ebXML *Message-Service* implementation that receives that *Message* to
2389 facilitate the elimination of duplicates. This duration also applies to response messages that are
2390 kept persistently to allow automatic replies to duplicate messages without their repeated
2391 processing by the application. For rules that govern the **PersistDuration** element, refer to
2392 Sections 8.4.23.4 and 8.4.41.2.

2393

2394 8.4.43 SenderNonRepudiation element

2395 The *SenderNonRepudiation element* conveys the message sender's requirements and certificate
 2396 for non-repudiation. Non-repudiation both proves who sent a *Message* and prevents later
 2397 repudiation of the contents of the *Message*. Non-repudiation is based on signing the *Message*
 2398 using XML Digital Signature[XMLDSIG]. The element structure is as follows:

```

2399
2400     <tp:SenderNonRepudiation>
2401         <tp:NonRepudiationProtocol>
2402             http://www.w3.org/2000/09/xmldsig#
2403         </tp:NonRepudiationProtocol>
2404         <tp:HashFunction>
2405             http://www.w3.org/2000/09/xmldsig#sha1
2406         </tp:HashFunction>
2407         <tp:SignatureAlgorithm>
2408             http://www.w3.org/2000/09/xmldsig#dsa-sha1
2409         </tp:SignatureAlgorithm>
2410         <tp:SigningCertificateRef tp:certId="CompanyA_SigningCert"/>
2411     </tp:SenderNonRepudiation>
2412
  
```

2413 If the *SenderNonRepudiation* element is omitted, the *Messages* are not digitally signed.

2414 The *SenderNonRepudiation* element is comprised of the following child elements:

- 2416 • a REQUIRED *NonRepudiationProtocol* element,
- 2417 • a REQUIRED *HashFunction* (e.g. SHA1, MD5) element,
- 2418 • a REQUIRED *SignatureAlgorithm* element,
- 2419 • a REQUIRED *SigningCertificateRef* element

2421 8.4.44 NonRepudiationProtocol element

2422 The REQUIRED *NonRepudiationProtocol* element identifies the technology that will be used to
 2423 digitally sign a *Message*. It has a single IMPLIED *version* attribute whose value is a string that
 2424 identifies the version of the specified technology.

2426 8.4.45 HashFunction element

2427 The REQUIRED *HashFunction* element identifies the algorithm that is used to compute the
 2428 digest of the *Message* being signed.

2430 8.4.46 SignatureAlgorithm element

2431 The REQUIRED *SignatureAlgorithm* element identifies the algorithm that is used to compute
 2432 the value of the digital signature. Expected values include: RSA-MD5, RSA-SHA1, DSA-MD5,
 2433 DSA-SHA1, SHA1withRSA, MD5withRSA, and so on.

2434
 2435 NOTE: Implementations should be prepared for values in upper and/or lower case and
 2436 with varying usage of hyphens and conjunctions.

2437
 2438 The *SignatureAlgorithm* element has three attributes:

- 2439 • an IMPLIED *oid* attribute,
- 2440 • an IMPLIED *w3c* attribute,
- 2441 • an IMPLIED *enumeratedType* attribute.

2442

2443 8.4.46.1 oid attribute

2444 The *oid* attribute serves as a way to supply an object identifier for the signature algorithm. The
 2445 formal definition of OIDs comes from ITU-T recommendation X.208 (ASN.1), chapter 28; the
 2446 assignment of the "top of the tree" is given in Appendix B, Appendix C and Appendix D of
 2447 X.208 (<http://www.itu.int/POD/>). Commonly used values (in the IETF dotted integer format) for
 2448 signature algorithms include:

- 2449 • 1.2.840.113549.1.1.4 - MD5 with RSA encryption,
- 2450 • 1.2.840.113549.1.1.5 - SHA-1 with RSA Encryption.

2451

2452 8.4.46.2 w3c attribute

2453 The *w3c* attribute serves as a way to supply an object identifier for the signature algorithm. The
 2454 definitions of these values are found in the [XMLDSIG] or [XMLENC] specifications. Expected
 2455 values for signature algorithms include:

- 2456 • <http://www.w3.org/2000/09/xmlldsig#dsa-sha1>,
- 2457 • <http://www.w3.org/2000/09/xmlldsig#rsa-sha1>.

2458

2459 8.4.46.3 enumeratedType attribute

2460 The *enumeratedType* attribute specifies a different way of interpreting the text value of the
 2461 *SignatureAlgorithm* element. This attribute is for identifying future signature algorithm
 2462 identification schemes and formats.

2463

2464 8.4.47 SigningCertificateRef element

2465 The REQUIRED *SigningCertificateRef* element identifies the certificate the sender uses for
 2466 signing messages. Its REQUIRED IDREF attribute, *certId* refers to the *Certificate* element
 2467 (under *PartyInfo*) that has the matching ID attribute value.

2468

2469 8.4.48 SenderDigitalEnvelope element

2470 The *SenderDigitalEnvelope* element provides the sender's requirements for message encryption
 2471 using the [DIGENV] digital-envelope method. Digital-envelope is a procedure in which the
 2472 *Message* is encrypted by symmetric encryption (shared secret key) and the secret key is sent to
 2473 the *Message* recipient encrypted with the recipient's public key. The element structure is:

2474

```

2475 <tp:SenderDigitalEnvelope>
2476   <tp:DigitalEnvelopeProtocol tp:version="2.0">
2477     S/MIME
2478   </tp:DigitalEnvelopeProtocol>
2479   <tp:EncryptionAlgorithm>DES-CBC</tp:EncryptionAlgorithm>
2480   <tp:EncryptionSecurityDetailsRef
2481     tp:securityId="CompanyA_MessageSecurity"/>
2482 </tp:SenderDigitalEnvelope>
2483 
```

2484

The *SenderDigitalEnvelope* element contains

Collaboration-Protocol Profile and Agreement Specification

60 of 156

Page

- 2485 • a REQUIRED `DigitalEnvelopeProtocol` element,
- 2486 • a REQUIRED *EncryptionAlgorithm* element
- 2487 • zero or one `EncryptionSecurityDetailsRef` element.
- 2488 •

2489 **8.4.49 DigitalEnvelopeProtocol element**

2490 The REQUIRED *DigitalEnvelopeProtocol* element identifies the message encryption protocol to
 2491 be used. The REQUIRED *version* attribute identifies the version of the protocol.
 2492

2493 **8.4.50 EncryptionAlgorithm element**

2494 The REQUIRED *EncryptionAlgorithm* element identifies the encryption algorithm to be used.
 2495 See also Section 8.4.32.

2496
 2497 The *EncryptionAlgorithm* element has four attributes:

- 2498 • an IMPLIED *minimumStrength* attribute,
- 2499 • an IMPLIED *oid* attribute,
- 2500 • an IMPLIED *w3c* attribute,
- 2501 • an IMPLIED *enumeratedType* attribute.
- 2502 •

2503 **8.4.50.1 minimumStrength attribute**

2504 The *minimumStrength* attribute describes the effective strength the encryption algorithm MUST
 2505 provide in terms of “effective” or random bits. This value is less than the key length in bits when
 2506 check bits are used in the key. So, for example, the 8 check bits of a 64-bit DES key would not
 2507 be included in the count, and to require a minimum strength the same as that supplied by DES
 2508 would be reported by setting *minimumStrength* to 56.
 2509

2510 **8.4.50.2 oid attribute**

2511 The *oid* attribute serves as a way to supply an object identifier for the encryption algorithm. The
 2512 formal definition of OIDs comes from ITU-T recommendation X.208 (ASN.1), chapter 28; the
 2513 assignment of the "top of the tree" is given in Appendix B, Appendix C and Appendix D of
 2514 X.208 (<http://www.itu.int/POD/>). Commonly used values (in the IETF dotted integer format) for
 2515 encryption algorithms include:

- 2516 • 1.2.840.113549.3.2 (RC2-CBC), 1.2.840.113549.3.4 (RC4 Encryption Algorithm),
- 2517 • 1.2.840.113549.3.7 (DES-EDE3-CBC), 1.2.840.113549.3.9 (RC5 CBC Pad),
- 2518 • 1.2.840.113549.3.10 (DES CDMF), 1.2.840.1.3.14.3.2.7 (DES-CBC).

2520 **8.4.50.3 w3c attribute**

2521 The *w3c* attribute serves as a way to supply an object identifier for the encryption algorithm. The
 2522 definitions of these values are in the [XMLENC] specification. Expected values include:

- 2523 • <http://www.w3.org/2001/04/xmlenc#3des-cbc>,
- 2524 • <http://www.w3.org/2001/04/xmlenc#aes128-cbc>,
- 2525 • <http://www.w3.org/2001/04/xmlenc#aes256-cbc>.

2526

2527 **8.4.50.4 enumeratedTypeAttribute**

2528 The *enumeratedType* attribute specifies a way of interpreting the text value of the
2529 *EncryptionAlgorithm* element. This attribute is for identifying future algorithm identification
2530 schemes and formats.
2531

2532 **8.4.51 EncryptionSecurityDetailsRef element**

2533 The *EncryptionSecurityDetailsRef* element identifies the trust anchors and security policy that
2534 this (sending) *Party* will apply to the other (receiving) *Party*'s encryption certificate. Its
2535 REQUIRED IDREF attribute, *securityId*, refers to the *SecurityDetails* element (under
2536 *PartyInfo*) that has the matching ID attribute value.
2537

2538 **8.4.52 NamespaceSupported element**

2539 The *NamespaceSupported* element may be included zero or more times. Each occurrence of the
2540 *NamespaceSupported* element identifies one namespace supported by the messaging service
2541 implementation. It has a REQUIRED *location* attribute and an IMPLIED *version* attribute. The
2542 *location* attribute supplies a URI for retrieval of the schema associated with the namespace. The
2543 *version* attribute provides a version value, when one exists, for the namespace. While the
2544 *NamespaceSupported* element can be used to list the namespaces that could be expected to be
2545 used during document exchange, the motivation is primarily for extensions, version variants, and
2546 other enhancements that might not be expected, or have only recently emerged into use.
2547

2548 For example, support for Security Assertion Markup Language[SAML] would be defined as
2549 follows:

```
2550  
2551 <tp:NamespaceSupported  
2552 tp:location="http://www.oasis-  
2553 open.org/committees/security/docs/draft-sstc-schema-  
2554 assertion-27.xsd" tp:version="1.0">  
2555 http://www.oasis-open.org/committees/security/docs/draft-  
2556 sstc-schema-assertion-27.xsd</tp:NamespaceSupported>  
2557
```

2558 In addition, the *NamespaceSupported* element can be used to identify the namespaces associated
2559 with the message body parts (see Section 8.5), and especially when these namespaces are not
2560 implicitly indicated through parts of the *ProcessSpecification* or when they indicate extensions
2561 of namespaces for payload body parts.
2562

2563 **8.4.53 ebXMLReceiverBinding element**

2564 The *ebXMLReceiverBinding* element describes properties related to receiving messages with the
2565 ebXML *Message Service*[ebMS]. The *ebXMLReceiverBinding* element is comprised of the
2566 following child elements:

- 2567 • zero or one *ReliableMessaging* element (see Section 8.4.41),
- 2568 • zero or one *ReceiverNonRepudiation* element which specifies the receiver's
2569 requirements for message signing,

- 2570 • zero or one **ReceiverDigitalEnvelope** element which specifies the receiver's requirements
 2571 and certificate for encryption by the digital-envelope[DIGENV] method,
 2572 • zero or more **NamespaceSupported** elements (see Section 8.4.52).

2573

2574 The **ebXMLReceiverBinding** element has one attribute:

- 2575 • a REQUIRED **version** attribute (see Section 8.4.40.1)

2576

2577 NOTE: A CPA could be valid even when omitting all children under
 2578 **ebXMLReceiverBinding**.

2579 8.4.54 ReceiverNonRepudiation element

2580 The **ReceiverNonRepudiation element** conveys the message receiver's requirements for non-
 2581 repudiation. Non-repudiation both proves who sent a *Message* and prevents later repudiation of
 2582 the contents of the *Message*. Non-repudiation is based on signing the *Message* using XML
 2583 Digital Signature[XMLDSIG]. The element structure is as follows:

2584

```
2585 <tp:ReceiverNonRepudiation>
2586   <tp:NonRepudiationProtocol>
2587     http://www.w3.org/2000/09/xmldsig#
2588   </tp:NonRepudiationProtocol>
2589   <tp:HashFunction>
2590     http://www.w3.org/2000/09/xmldsig#sha1
2591   </tp:HashFunction>
2592   <tp:SignatureAlgorithm>
2593     http://www.w3.org/2000/09/xmldsig#dsa-sha1
2594   </tp:SignatureAlgorithm>
2595   <tp:SigningSecurityDetailsRef
2596     tp:securityId="CompanyA_MessageSecurity"/>
2597 </tp:ReceiverNonRepudiation>
```

2598

2599 If the **ReceiverNonRepudiation** element is omitted, the *Messages* are not digitally signed.

2600

2601 The **ReceiverNonRepudiation** element is comprised of the following child elements:

- 2602 • a REQUIRED **NonRepudiationProtocol** element (see Section 8.4.44),
 2603 • a REQUIRED **HashFunction** (e.g. SHA1, MD5) element (see Section 8.4.45),
 2604 • a REQUIRED **SignatureAlgorithm** element (see Section 8.4.46),
 2605 • zero or one **SigningSecurityDetailsRef** element

2606

2607 8.4.55 SigningSecurityDetailsRef element

2608 The **SigningSecurityDetailsRef** element identifies the trust anchors and security policy that this
 2609 (receiving) *Party* will apply to the other (sending) *Party*'s signing certificate. Its REQUIRED
 2610 IDREF attribute, **securityId**, refers to the **SecurityDetails** element (under **PartyInfo**) that has the
 2611 matching ID attribute value.

2612

2613 8.4.56 ReceiverDigitalEnvelope element

2614 The **ReceiverDigitalEnvelope** element provides the receiver's requirements for message
 2615 encryption using the [DIGENV] digital-envelope method. Digital-envelope is a procedure in

2616 which the *Message* is encrypted by symmetric encryption (shared secret key) and the secret key
 2617 is sent to the *Message* recipient encrypted with the recipient's public key. The element structure
 2618 is:

```
2619
2620     <tp:ReceiverDigitalEnvelope>
2621         <tp:DigitalEnvelopeProtocol tp:version="2.0">
2622             S/MIME
2623         </tp:DigitalEnvelopeProtocol>
2624         <tp:EncryptionAlgorithm>DES-CBC</tp:EncryptionAlgorithm>
2625         <tp:EncryptionCertificateRef
2626             tp:certId="CompanyA_EncryptionCert"/>
2627     </tp:ReceiverDigitalEnvelope>
2628
```

2629 The *ReceiverDigitalEnvelope* element contains

- 2630 • a REQUIRED *DigitalEnvelopeProtocol* element (see Section 8.4.49),
- 2631 • a REQUIRED *EncryptionAlgorithm* element (see Section 8.4.50),
- 2632 • a REQUIRED *EncryptionCertificateRef* element.

2633

2634 8.4.57 EncryptionCertificateRef element

2635 The REQUIRED *EncryptionCertificateRef* element identifies the certificate the sender uses for
 2636 encrypting messages. Its REQUIRED IDREF attribute, *certId* refers to the *Certificate* element
 2637 (under *PartyInfo*) that has the matching ID attribute value.

2638 .

2639 8.4.58 OverrideMshActionBinding element

2640 The *OverrideMshActionBinding* element can occur zero or more times. It has two REQUIRED
 2641 attributes. The *action* attribute identifies the *Message Service Handler* level action whose
 2642 delivery is not to use the default *DeliveryChannel* for *Message Service Handler* actions. The
 2643 *channelId* attribute specifies the *DeliveryChannel* to be used instead.

2644

2645 8.5 SimplePart element

2646 The *SimplePart* element provides a repeatable list of the constituent parts, primarily identified by
 2647 the MIME content-type value. The *SimplePart* element has two REQUIRED attributes: *id* and
 2648 *mimetype*. The *id* attribute, of type ID, provides the value that will be used later to reference this
 2649 *Message* part when specifying how the parts are packaged into composites, if composite
 2650 packaging is present. The *mimetype* attribute can provide actual values of content-type for the
 2651 simple *Message* part being specified. The attribute's values may also make use of an asterisk
 2652 wildcard, "*", to indicate either an arbitrary top-level type, an arbitrary sub-type, or a completely
 2653 arbitrary type, "*/*". SimpleParts with wildcards in types can be used in indicating more open
 2654 packaging processing capabilities.

2655

2656 *SimplePart* has an IMPLIED *mimparameters* attribute, whose use is described in section 8.6.2.
 2657 *SimplePart* also has an IMPLIED *xlink:role* attribute which identifies some resource that
 2658 describes the mime part or its purpose; see Appendix F for a discussion of the use of this value
 2659 within [ebMS]. If present, then it SHALL have a value that is a valid URI in accordance with the
 2660 [XLINK] specification.

2661

2662 The following are examples of *SimplePart* elements:

2663

```
2664 <tp:SimplePart tp:id="I001" tp:mimetype="text/xml"/>
```

```
2665 <tp:SimplePart tp:id="I002" tp:mimetype="application/xml"/>
```

```
2666 <tp:SimplePart tp:id="I002" tp:mimetype="*/xml"/>
```

2667

2668 The *SimplePart* element can have zero or more *NamespaceSupported* elements. Each of these
2669 identifies any namespace supported for the XML that is packaged in the parent simple body part.

2670

2671 The context of *Packaging* can very easily render it pointless to list all the namespaces used in a
2672 *SimplePart*. For example, when defining the *SimplePart* for a SOAP envelope, as part of an
2673 ebXML Message, it is not necessary to list all the namespaces. If, however, any unusual
2674 extensions, new versions, or unusual security extensions are present, it is useful to announce
2675 these departures explicitly in the packaging. It is not, however, incorrect to list all namespaces
2676 used in a *SimplePart*, even where these namespaces have been mandated by a given messaging
2677 protocol. By convention, when a full listing of namespaces is supplied within a *SimplePart*
2678 element, the first *NamespaceSupported* element identifies the schema for the *SimplePart* while
2679 subsequent *NamespaceSupported* elements represent namespaces that are imported by that
2680 schema. Any additional *NamespaceSupported* elements indicate extensions.

2681

2682 NOTE: The explicit identification of imported namespaces is discretionary. Thus, the
2683 CPP and CPA examples in Appendix A and Appendix B explicitly identify the ebXML
2684 Messaging Service namespace but omit the SOAP envelope and XML Digital Signature
2685 namespaces that are imported into the schema for the ebXML Messaging Service
2686 namespace.

2687

2688 The same *SimplePart* element can be referenced from (i.e., reused in) multiple *Packaging*
2689 elements.

2690

2691 8.6 Packaging element

2692 The subtree of the *Packaging* element provides specific information about how the *Message*
2693 *Header* and payload constituent(s) are packaged for transmittal over the transport, including the
2694 crucial information about what document-level security packaging is used and the way in which
2695 security features have been applied. Typically the subtree under the *Packaging* element indicates
2696 the specific way in which constituent parts of the *Message* are organized. MIME processing
2697 capabilities are typically the capabilities or agreements described in this subtree. The *Packaging*
2698 element provides information about MIME content types, XML namespaces, security
2699 parameters, and MIME structure of the data that is exchanged between *Parties*.

2700

2701 The following is an example of a *Packaging* element which references the example *SimplePart*
2702 elements given in Section 8.5:

2703

```
2704 <!-- Simple ebXML S/MIME Packaging for application-based payload
```

```
2705 encryption -->
```

```
2706 <tp:Packaging>
```

```

2707     <tp:ProcessingCapabilities tp:generate="true" tp:parse="true"/>
2708     <tp:CompositeList>
2709         <tp:Encapsulation
2710             <!-- I002 is the payload being encrypted -->
2711             tp:id="I003"
2712             tp:mimetype="application/pkcs7-mime"
2713             tp:mimeparameters="smime-type=&quot;enveloped-data&quot;">
2714             <Constituent tp:idref="I002"/>
2715         </tp:Encapsulation>
2716         <tp:Composite tp:id="I004"
2717             <!-- I001 is the SOAP envelope. The ebXML message is made
2718             up of the SOAP envelope and the encrypted payload. -->
2719             tp:mimetype="multipart/related"
2720             tp:mimeparameters="type=&quot;text/xml&quot;
2721             version=&quot;1.0&quot;">
2722             <tp:Constituent tp:idref="I001"/>
2723             <tp:Constituent tp:idref="I003"/>
2724         </tp:Composite>
2725     </tp:CompositeList>
2726 </tp:Packaging>
2727

```

2728 The **Packaging** element has one attribute; the REQUIRED *id* attribute, with type ID. It is
 2729 referred to in the **ThisPartyActionBinding** element, by using the IDREF attribute, *packageId*.

2730
 2731 The child elements of the **Packaging** element are **ProcessingCapabilities** and **CompositeList**.
 2732 This set of elements can appear one or more times as a child of each **Packaging** element.
 2733

2734 8.6.1 ProcessingCapabilities element

2735 The **ProcessingCapabilities** element has two REQUIRED attributes with Boolean values of
 2736 either "true" or "false". The attributes are *parse* and *generate*. Normally, these attributes will
 2737 both have values of "true" to indicate that the packaging constructs specified in the other child
 2738 elements can be both produced as well as processed at the software *Message* service layer.
 2739 At least one of the *generate* or *parse* attributes MUST be true.
 2740

2741 8.6.2 CompositeList element

2742 The final child element of **Packaging** is **CompositeList**, which is a container for the specific way
 2743 in which the simple parts are combined into groups (MIME multipart) or encapsulated within
 2744 security-related MIME content-types. The **CompositeList** element SHALL be omitted from
 2745 **Packaging** when no security encapsulations or composite multipart are used. When the
 2746 **CompositeList** element is present, the content model for the **CompositeList** element is a
 2747 repeatable sequence of choices of **Composite** or **Encapsulation** elements. The **Composite** and
 2748 **Encapsulation** elements can appear intermixed as desired. The sequence in which the choices
 2749 are presented is important because, given the recursive character of MIME packaging,
 2750 composites or encapsulations can include previously mentioned composites (or rarely,
 2751 encapsulations) in addition to the *Message* parts characterized within the **SimplePart** subtree.
 2752 Therefore, the "top-level" packaging will be described last in the sequence.
 2753

2754 The **Composite** element has the following attributes:

- 2755 • a REQUIRED *mimetype* attribute,

- 2756 • a REQUIRED *id* attribute,
- 2757 • an IMPLIED *mimeparameters* attribute.

2758
2759 The *mimetype* attribute provides the value of the MIME content-type for this *Message* part, and
2760 this will be some MIME composite type, such as "multipart/related" or "multipart/signed". The
2761 *id* attribute, type ID, provides a way to refer to this composite if it needs to be mentioned as a
2762 constituent of some later element in the sequence. The *mimeparameters* attribute provides the
2763 values of any significant MIME parameter (such as "type=application/xml") that is needed to
2764 understand the processing demands of the content-type.

2765
2766 The *Composite* element has one child element, *Constituent*.

2767
2768 The *Constituent* element has one REQUIRED attribute, *idref* of type IDREF, an IMPLIED
2769 boolean attribute *excludeFromSignature*, and two IMPLIED nonNegativeInteger attributes,
2770 *minOccurs* and *maxOccurs*.

2771
2772 The *idref* attribute has as its value the value of the *id* attribute of a previous *Composite*,
2773 *Encapsulation*, or *SimplePart* element. The purpose of this sequence of *Constituents* is to
2774 indicate both the contents and the order of what is packaged within the current *Composite* or
2775 *Encapsulation*.

2776
2777 The *excludeFromSignature* attribute indicates that this *Constituent* is not to be included as part
2778 of the ebXML message [XMLDSIG] signature. In other words, the signature generated by the
2779 *Message Service Handler* should not include a *ds:Reference* element to provide a digest for this
2780 *Constituent* of the *Message*. This attribute is applicable only if the *Constituent* is part of the top-
2781 level *Composite* that corresponds to the entire ebXML *Message*.

2782
2783 The *minOccurs* and *maxOccurs* attributes serve to specify the value or range of values that the
2784 referred to item may occur within *Composite*. When unused, it is understood that the item is used
2785 exactly once.

2786
2787 The *Encapsulation* element is typically employed to indicate the use of MIME security
2788 mechanisms, such as [S/MIME] or Open-PGP[RFC2015]. A security body part can encapsulate a
2789 MIME part that has been previously characterized. For convenience, all such security structures
2790 are under the *Encapsulation* element, even when technically speaking the data is not "inside" the
2791 body part. (In other words, the so-called clear-signed or detached signature structures possible
2792 with MIME multipart/signed are for simplicity found under the *Encapsulation* element.)

2793
2794 Another possible use of the *Encapsulation* element is to represent the application of a
2795 compression algorithm such as gzip [ZLIB] to some part of the payload, prior to its being
2796 encrypted and or signed.

2797
2798 The *Encapsulation* element has the following attributes:

- 2799 • a REQUIRED *mimetype* attribute,
- 2800 • a REQUIRED *id* attribute,

- 2801 • an IMPLIED *mimeparameters* attribute.

2802
 2803 The *mimetype* attribute provides the value of the MIME content-type for this *Message* part, such
 2804 as "application/pkcs7-mime". The *id* attribute, type ID, provides a way to refer to this
 2805 encapsulation if it needs to be mentioned as a constituent of some later element in the sequence.
 2806 The *mimeparameters* attribute provides the values of any significant MIME parameter(s)
 2807 needed to understand the processing demands of the content-type.

2808
 2809 Both the *Encapsulation* element and the *Composite* element have child elements consisting of a
 2810 *Constituent* element or of a repeatable sequence of *Constituent* elements, respectively.

2811
 2812 The *Constituent* element also has zero or one *SignatureTransform* child element and zero or
 2813 one *EncryptionTransform* child element. The *SignatureTransform* element is intended for use
 2814 with XML Digital Signature [XMLDSIG]. When present, it identifies the transforms that must
 2815 be applied to the source data before a digest is computed. The *EncryptionTransform* element is
 2816 intended for use with XML Encryption [XMLENC]. When present, it identifies the transforms
 2817 that must be applied to a *CipherReference* before decryption can be performed. The
 2818 *SignatureTransforms* element and the *EncryptionTransforms* element each contains one or
 2819 more *ds:Transform* [XMLDSIG] elements.

2820

2821 8.7 Signature element

2822 The *Signature* element (cardinality zero or one) enables the CPA to be digitally signed using
 2823 technology that conforms with the XML Digital Signature specification[XMLDSIG]. The
 2824 *Signature* element is the root of a subtree of elements used for signing the *CPP*. The syntax is:

2825
 2826 `<tp:Signature>...</tp:Signature>`
 2827

2828 The *Signature* element contains one or more *ds:Signature* elements. The content of the
 2829 *ds:Signature* element and any sub-elements are defined by the XML Digital Signature
 2830 specification. See Section 9.9 for a detailed discussion.

2831
 2832 NOTE: It is necessary to wrap the *ds:Signature* elements with a *Signature* element in the
 2833 target namespace to allow for the possibility of having wildcard elements (with
 2834 namespace="##other") within the *CollaborationProtocolProfile* and
 2835 *CollaborationProtocolAgreement* elements. The content model would be ambiguous
 2836 without the wrapping.

2837

2838 The following additional constraints on *ds:Signature* are imposed:

2839

- 2840 • A *CPP* MUST be considered invalid if any *ds:Signature* element fails core validation as
 2841 defined by the XML Digital Signature specification[XMLDSIG].

2842

- 2843
- 2844
- 2845
- 2846
- Whenever a *CPP* is signed, each *ds:Reference* element within a *ProcessSpecification* element MUST pass reference validation and each *ds:Signature* element MUST pass core validation.

2847

2848

2849

NOTE: In case a *CPP* is unsigned, software might nonetheless validate the *ds:Reference* elements within *ProcessSpecification* elements and report any exceptions.

2850

2851

2852

2853

NOTE: Software for creation of *CPPs* and *CPAs* MAY recognize *ds:Signature* and automatically insert the element structure necessary to define signing of the *CPP* and *CPA*. Signature generation is outlined in Section 9.9.1.1; details of the cryptographic process are outside the scope of this specification.

2854

2855

2856

2857

NOTE: See non-normative note in Section 8.4.4.5 for a discussion of times at which validity tests MAY be made.

2858 **8.8 Comment element**

2859

2860

2861

2862

2863

2864

The *CollaborationProtocolProfile* element contains zero or more *Comment* elements. The *Comment* element is a textual note that can be added to serve any purpose the author desires. The language of the *Comment* is identified by a REQUIRED *xml:lang* attribute. The *xml:lang* attribute MUST comply with the rules for identifying languages specified in [XML]. If multiple *Comment* elements are present, each can have a different *xml:lang* attribute value. An example of a *Comment* element follows:

2865

2866

```
<tp:Comment xml:lang="en-US">This is a CPA between A and B</tp:Comment>
```

2867

2868

2869

When a *CPA* is composed from two *CPPs*, all *Comment* elements from both *CPPs* SHALL be included in the *CPA* unless the two *Parties* agree otherwise.

2870 **9 CPA Definition**

2871 A *Collaboration-Protocol Agreement (CPA)* defines the capabilities that two *Parties* need to
 2872 agree upon to enable them to engage in electronic *Business* for the purposes of the particular
 2873 *CPA*. This section defines and discusses the details of the *CPA*. The discussion is illustrated with
 2874 some XML fragments.

2875
 2876 Most of the XML elements in this section are described in detail in Section 8, "CPP Definition".
 2877 In general, this section does not repeat that information. The discussions in this section are
 2878 limited to those elements that are not in the *CPP* or for which additional discussion is needed in
 2879 the *CPA* context. See also Appendix D for the XML Schema, and Appendix B for an example of
 2880 a *CPA* document.

2881

2882 **9.1 CPA Structure**

2883 Following is the overall structure of the *CPA*:

2884

```

2885     <CollaborationProtocolAgreement
2886         xmlns:tp="http://www.oasis-open.org/committees/ebxml-
2887 cppa/schema/cpp-cpa-2_0.xsd"
2888         xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
2889         xmlns:xlink="http://www.w3.org/1999/xlink"
2890         tp:cpaid="YoursAndMyCPA"
2891         tp:version="2.0a">
2892         <tp:Status tp:value="proposed"/>
2893         <tp:Start>1988-04-07T18:39:09</Start>
2894         <tp:End>1990-04-07T18:40:00</End>
2895         <!-- ConversationConstraints MAY appear 0 or 1 time -->
2896         <tp:ConversationConstraints
2897             tp:invocationLimit="100"
2898             tp:concurrentConversations="4"/>
2899         <tp:PartyInfo>
2900             ...
2901         </tp:PartyInfo>
2902         <tp:PartyInfo>
2903             ...
2904         </tp:PartyInfo>
2905         <tp:SimplePart tp:id="..."> <!-- one or more -->
2906             ...
2907         </tp:SimplePart>
2908         <tp:Packaging tp:id="..."> <!-- one or more -->
2909             ...
2910         </tp:Packaging>
2911         <tp:Signature> <!-- zero or one time -->
2912             ...
2913         </tp:Signature>
2914         <tp:Comment xml:lang="en-GB">any text</Comment> <!--
2915 zero or more -->
2916     </tp:CollaborationProtocolAgreement>
  
```

2917

2918 9.2 CollaborationProtocolAgreement element

2919 The *CollaborationProtocolAgreement* element is the root element of a *CPA*. It has a
 2920 REQUIRED *cpaid* attribute that supplies a unique identifier for the document. The value of the
 2921 *cpaid* attribute SHALL be assigned by one *Party* and used by both. It is RECOMMENDED that
 2922 the value of the *cpaid* attribute be a URI. The value of the *cpaid* attribute SHALL be used as the
 2923 value of the *CPAId* element in the ebXML *Message Header*[ebMS] or of a similar element in a
 2924 *Message Header* of an alternative messaging service.

2925

2926 NOTE: Each *Party* might associate a local identifier with the *cpaid* attribute.

2927

2928 In addition, the *CollaborationProtocolAgreement* element has a REQUIRED *version* attribute.
 2929 This attribute indicates the version of the schema to which the *CPA* conforms. The value of the
 2930 *version* attribute SHOULD be a string such as "2_0a", "2_0b", etc.

2931

2931 NOTE: The method of assigning unique *cpaid* values is left to the implementation.

2932

2933 The *CollaborationProtocolAgreement* element has REQUIRED [XML] Namespace[XMLNS]
 2934 declarations that are defined in Section 8, "CPP Definition".

2935

2936 The *CollaborationProtocolAgreement* element is comprised of the following child elements,
 2937 most of which are described in greater detail in subsequent sections:

- 2938 • a REQUIRED *Status* element that identifies the state of the process that creates the *CPA*,
- 2939 • a REQUIRED *Start* element that records the date and time that the *CPA* goes into effect,
- 2940 • a REQUIRED *End* element that records the date and time after which the *CPA* MUST be
 2941 renegotiated by the *Parties*,
- 2942 • zero or one *ConversationConstraints* element that documents certain agreements about
 2943 conversation processing,
- 2944 • two REQUIRED *PartyInfo* elements, one for each *Party* to the *CPA*,
- 2945 • one or more *SimplePart* elements,
- 2946 • one or more *Packaging* elements,
- 2947 • zero or one *Signature* element that provides for signing of the *CPA* using the XML
 2948 Digital Signature[XMLDSIG] standard,
- 2949 • zero or more *Comment* elements.

2950

2951 9.3 Status Element

2952 The *Status* element records the state of the composition/negotiation process that creates the *CPA*.
 2953 An example of the *Status* element follows:

2954

```
2955 <tp:Status tp:value="proposed"/>
```

2956

2957 The *Status* element has a REQUIRED *value* attribute that records the current state of
 2958 composition of the *CPA*. This attribute is an enumeration comprised of the following possible
 2959 values:

- 2960 • "proposed", meaning that the *CPA* is still being negotiated by the *Parties*,
2961 • "agreed", meaning that the contents of the *CPA* have been agreed to by both *Parties*,
2962 • "signed", meaning that the *CPA* has been "signed" by one or more of the *Parties*. This
2963 "signing" takes the form of a digital signature that is described in Section 9.7 below.
2964

2965 NOTE: The *Status* element MAY be used by a *CPA* composition and negotiation tool to
2966 assist it in the process of building a *CPA*.
2967

2968 NOTE: The value of the *Status* element's *value* attribute is set to "signed" before the first
2969 *Party* signs. Even though excluding *value* attribute from a signature might be technically
2970 feasible, it is preferable to change the attribute's value to "signed" prior to the first
2971 signature, and maintain it as "signed" for any subsequent signatures.
2972

2973 9.4 CPA Lifetime

2974 The lifetime of the *CPA* is given by the *Start* and *End* elements. The syntax is:

```
2975 <tp:Start>1988-04-07T18:39:09Z</tp:Start>  
2976 <tp:End>1990-04-07T18:40:00Z</tp:End>  
2977  
2978
```

2979 9.4.1 Start element

2980 The *Start* element specifies the starting date and time of the *CPA*. The *Start* element SHALL be
2981 a string value that conforms to the content model of a canonical dateTime type as defined in the
2982 XML Schema Datatypes Specification[XMLSCHEMA-2]. For example, to indicate 1:20 pm
2983 UTC (Coordinated Universal Time) on May 31, 1999, a *Start* element would have the following
2984 value:

```
2985 1999-05-31T13:20:00Z  
2986  
2987
```

2988 The *Start* element SHALL be represented as Coordinated Universal Time (UTC).
2989

2990 9.4.2 End element

2991 The *End* element specifies the ending date and time of the *CPA*. The *End* element SHALL be a
2992 string value that conforms to the content model of a canonical dateTime type as defined in the
2993 XML Schema Datatypes Specification[XMLSCHEMA-2]. For example, to indicate 1:20 pm
2994 UTC (Coordinated Universal Time) on May 31, 1999, an *End* element would have the following
2995 value:

```
2996 1999-05-31T13:20:00Z  
2997  
2998
```

2999 The *End* element SHALL be represented as Coordinated Universal Time (UTC).
3000

3001 When the end of the *CPA's* lifetime is reached, any *Business Transactions* that are still in
3002 progress SHALL be allowed to complete and no new *Business Transactions* SHALL be started.

3003 When all in-progress *Business Transactions* on each conversation are completed, the
3004 *Conversation* SHALL be terminated whether or not it was completed.

3005
3006 When a *CPA* is signed, software for signing the agreements SHALL warn if any signing
3007 certificate's validity expires prior to the proposed time for ending the *CPA*. The opportunity to
3008 renegotiate a *CPA End* value or to in some other way align certificate validity periods with *CPA*
3009 validity periods SHALL be made available. (Other ways to align these validity periods would
3010 include reissuing the signing certificates for a longer period or obtaining new certificates for this
3011 purpose.)

3012
3013 Signing software SHOULD also attempt to align the validity periods of certificates referred to
3014 within the *CPA* that perform security functions so as to not expire before the *CPA* expires. This
3015 alignment can occur in several ways including making use of *ds:KeyInfo*'s content model
3016 *ds:RetrievalMethod* so that a new certificate can be installed and still be retrieved in accordance
3017 with the information in *ds:RetrievalMethod*. If no alignment can be attained, signing software
3018 MUST warn the user of the situation that the *CPA* validity exceeds the validity of some of the
3019 certificates referred to within the *CPA*.

3020
3021 NOTE: If a *Business* application defines a conversation as consisting of multiple *Business*
3022 *Transactions*, such a conversation MAY be terminated with no error indication when the
3023 end of the lifetime is reached. The run-time system could provide an error indication to
3024 the application.

3025
3026 NOTE: It might not be feasible to wait for outstanding conversations to terminate before
3027 ending the *CPA* since there is no limit on how long a conversation can last.

3028
3029 NOTE: The run-time system SHOULD return an error indication to both *Parties* when a
3030 new *Business Transaction* is started under this *CPA* after the date and time specified in
3031 the *End* element.

3032

3033 9.5 ConversationConstraints Element

3034 The *ConversationConstraints* element places limits on the number of conversations under the
3035 *CPA*. An example of this element follows:

3036
3037

```
<tp:ConversationConstraints tp:invocationLimit="100"
3038 tp:concurrentConversations="4" />
```

3039
3040 The *ConversationConstraints* element has the following attributes:

- 3041 • an IMPLIED *invocationLimit* attribute,
- 3042 • an IMPLIED *concurrentConversations* attribute.

3043

3044 9.5.1 invocationLimit attribute

3045 The *invocationLimit* attribute defines the maximum number of conversations that can be
3046 processed under the *CPA*. When this number has been reached, the *CPA* is terminated and

3047 MUST be renegotiated. If no value is specified, there is no upper limit on the number of
3048 conversations and the lifetime of the *CPA* is controlled solely by the **End** element.
3049

3050 NOTE: The **invocationLimit** attribute sets a limit on the number of units of *Business* that
3051 can be performed under the *CPA*. It is a *Business* parameter, not a performance
3052 parameter. A *CPA* expires whichever terminating condition (**End** or **invocationLimit**) is
3053 first reached.
3054

3055 9.5.2 concurrentConversations attribute

3056 The **concurrentConversations** attribute defines the maximum number of conversations that can
3057 be in process under this *CPA* at the same time. If no value is specified, processing of concurrent
3058 conversations is strictly a local matter.
3059

3060 NOTE: The **concurrentConversations** attribute provides a parameter for the *Parties* to
3061 use when it is necessary to limit the number of conversations that can be concurrently
3062 processed under a particular *CPA*. For example, the back-end process might only support
3063 a limited number of concurrent conversations. If a request for a new conversation is
3064 received when the maximum number of conversations allowed under this *CPA* is already
3065 in process, an implementation MAY reject the new conversation or MAY enqueue the
3066 request until an existing conversation ends. If no value is given for
3067 **concurrentConversations**, how to handle a request for a new conversation for which
3068 there is no capacity is a local implementation matter.
3069

3070 9.6 PartyInfo Element

3071 The general characteristics of the **PartyInfo** element are discussed in Section 8.4.
3072

3073 The *CPA* SHALL have one **PartyInfo** element for each *Party* to the *CPA*. The **PartyInfo**
3074 element specifies the *Parties'* agreed terms for engaging in the *Business Collaborations* defined
3075 by the *Process-Specification* documents referenced by the *CPA*. If a *CPP* has more than one
3076 **PartyInfo** element, the appropriate **PartyInfo** element SHALL be selected from each *CPP* when
3077 composing a *CPA*.
3078

3079 In the *CPA*, there SHALL be one or more **PartyId** elements under each **PartyInfo** element. The
3080 values of these elements are the same as the values of the **PartyId** elements in the ebXML
3081 *Message Service* specification[ebMS] or similar messaging service specification. These **PartyId**
3082 elements SHALL be used within a **To** or **From Header** element of an ebXML *Message*.
3083

3084 9.6.1 ProcessSpecification element

3085 The **ProcessSpecification** element identifies the *Business Collaboration* that the two *Parties*
3086 have agreed to perform. There can be one or more **ProcessSpecification** elements in a *CPA*.
3087 Each SHALL be a child element of a separate **CollaborationRole** element. See the discussion in
3088 Section 8.4.3.
3089

3090 **9.7 SimplePart element**

3091 The *CollaborationProtocolAgreement* element SHALL contain one or more *SimplePart*
3092 elements. See Section 8.5 for details of the syntax of the *SimplePart* element.
3093

3094 **9.8 Packaging element**

3095 The *CollaborationProtocolAgreement* element SHALL contain one or more *Packaging*
3096 elements. See Section 8.6 for details of the syntax of the *Packaging* element.
3097

3098 **9.9 Signature element**

3099 A *CPA* document can be digitally signed by one or more of the *Parties* as a means of ensuring its
3100 integrity as well as a means of expressing the agreement just as a corporate officer's signature
3101 would do for a paper document. If signatures are being used to digitally sign an ebXML *CPA* or
3102 *CPP* document, then [XMLDSIG] SHALL be used to digitally sign the document.
3103

3104 The *Signature* element, if present, is made up of one to three *ds:Signature* elements. The *CPA*
3105 can be signed by one or both *Parties*. It is RECOMMENDED that both *Parties* sign the *CPA*.
3106 For signing by both *Parties*, one *Party* initially signs. The other *Party* then signs over the first
3107 *Party's* signature. The resulting *CPA* MAY then be signed by a notary.
3108

3109 The *ds:Signature* element is the root of a subtree of elements used for signing the *CPP*.
3110

3111 The content of this element and any sub-elements are defined by the XML Digital Signature
3112 specification[XMLDSIG]. The following additional constraints on *ds:Signature* are imposed:
3113

- 3114 • A *CPA* MUST be considered invalid if any *ds:Signature* fails core validation as defined
3115 by the XML Digital Signature specification.
3116
- 3117 • Whenever a *CPA* is signed, each *ds:Reference* within a *ProcessSpecification* MUST
3118 pass reference validation and each *ds:Signature* MUST pass core validation.
3119

3120 NOTE: In case a *CPA* is unsigned, software MAY nonetheless validate the *ds:Reference*
3121 elements within *ProcessSpecification* elements and report any exceptions.
3122

3123 Software for creation of *CPPs* and *CPAs* SHALL recognize *ds:Signature* and automatically
3124 insert the element structure necessary to define signing of the *CPP* and *CPA*. Signature creation
3125 itself is a cryptographic process that is outside the scope of this specification.
3126

3127 NOTE: See non-normative note in Section 8.4.4.5 for a discussion of times at which a
3128 *CPA* MAY be validated.
3129

3130 9.9.1 Persistent Digital Signature

3131 If [XMLDSIG] is used to sign an ebXML *CPP* or *CPA*, the process defined in this section of the
3132 specification SHALL be used.

3133

3134 9.9.1.1 Signature Generation

3135 Following are the steps to create a digital signature:

3136

- 3137 1. Create a *SignedInfo* element, a child element of *ds:Signature*. *SignedInfo* SHALL have
3138 child elements *SignatureMethod*, *CanonicalizationMethod*, and *Reference* as prescribed by
3139 [XMLDSIG].
- 3140 2. Canonicalize and then calculate the *SignatureValue* over *SignedInfo* based on algorithms
3141 specified in *SignedInfo* as specified in [XMLDSIG].
- 3142 3. Construct the *Signature* element that includes the *SignedInfo*, *KeyInfo*
3143 (RECOMMENDED), and *SignatureValue* elements as specified in [XMLDSIG].
- 3144 4. Include the namespace qualified *Signature* element in the document just signed, following
3145 the last *PartyInfo* element.

3146

3147 9.9.1.2 ds:SignedInfo element

3148 The *ds:SignedInfo* element SHALL be comprised of zero or one *ds:CanonicalizationMethod*
3149 element, the *ds:SignatureMethod* element, and one or more *ds:Reference* elements.

3150

3151 9.9.1.3 ds:CanonicalizationMethod element

3152 The *ds:CanonicalizationMethod* element as defined in [XMLDSIG], can occur zero or one
3153 time, meaning that the element need not appear in an instance of a *ds:SignedInfo* element. The
3154 default canonicalization method that is applied to the data to be signed is [XMLC14N] in the
3155 absence of a *ds:CanonicalizationMethod* element that specifies otherwise. This default SHALL
3156 also serve as the default canonicalization method for the ebXML *CPP* and *CPA* documents.

3157

3158 9.9.1.4 ds:SignatureMethod element

3159 The *ds:SignatureMethod* element SHALL be present and SHALL have an *Algorithm* attribute.
3160 The RECOMMENDED value for the *Algorithm* attribute is:

3161

3162 "http://www.w3.org/2000/09/xmlsig#sha1"

3163

3164 This RECOMMENDED value SHALL be supported by all compliant ebXML *CPP* or *CPA*
3165 software implementations.

3166

3167 9.9.1.5 ds:Reference element

3168 The *ds:Reference* element for the *CPP* or *CPA* document SHALL have a REQUIRED URI
3169 attribute value of "" to provide for the signature to be applied to the document that contains the
3170 *ds:Signature* element (the *CPA* or *CPP* document). The *ds:Reference* element for the *CPP* or
3171 *CPA* document can include an IMPLIED *type* attribute that has a value of:

3172

3173 "http://www.w3.org/2000/09/xmlsig#Object"

3174

3175 in accordance with [XMLDSIG]. This attribute is purely informative. It MAY be omitted.
 3176 Implementations of software designed to author or process an ebXML *CPA* or *CPP* document
 3177 SHALL be prepared to handle either case. The *ds:Reference* element can include the *id* attribute,
 3178 type ID, by which this *ds:Reference* element is referenced from a *ds:Signature* element.

3179

3180 9.9.1.6 ds:Transform element

3181 The *ds:Reference* element for the *CPA* or *CPP* document SHALL include a descendant
 3182 *ds:Transform* element that excludes the containing *ds:Signature* element and all its descendants.
 3183 This exclusion is achieved by means of specifying the *ds:Algorithm* attribute of the *Transform*
 3184 element as

```
3185     "http://www.w3.org/2000/09/xmlsig#enveloped-signature"
```

3186

3187 For example:

```
3188     <ds:Reference ds:URI="">
3189         <ds:Transforms>
3190             <ds:Transform
3191 ds:Algorithm="http://www.w3.org/2000/09/xmlsig#enveloped-
3192 signature"/>
3193         </ds:Transforms>
3194         <ds:DigestMethod
3195 ds:Algorithm="http://www.w3.org/2000/09/xmlsig#sha1"/>
3196         <ds:DigestValue>...</ds:DigestValue>
3197     </ds:Reference>
```

3198

3199 9.9.1.7 ds:Algorithm attribute

3200 The *ds:Transform* element SHALL include a *ds:Algorithm* attribute that has a value of:

3201

```
3202     http://www.w3.org/2000/09/xmlsig#enveloped-signature
```

3203

3204 NOTE: When digitally signing a *CPA*, it is RECOMMENDED that each *Party* sign the
 3205 document in accordance with the process described above.

3206

3207 When the two *Parties* sign the *CPA*, the first *Party* that signs the *CPA* SHALL sign only the
 3208 *CPA* contents, excluding their own signature. The second *Party* SHALL sign over the contents of
 3209 the *CPA* as well as the *ds:Signature* element that contains the first *Party's* signature. If
 3210 necessary, a notary can then sign over both signatures.

3211

3212 9.10 Comment element

3213 The *CollaborationProtocolAgreement* element contains zero or more *Comment* elements. See
 3214 Section 8.8 for details of the syntax of the *Comment* element.

3215

3216 9.11 Composing a CPA from Two CPPs

3217 This section discusses normative issues in composing a *CPA* from two *CPPs*. See also Appendix
 3218 E, "CPA Composition (Non-Normative)".

3219

3220 9.11.1 ID Attribute Duplication

3221 In composing a *CPA* from two *CPPs*, there is a hazard that ID attributes from the two *CPPs*
 3222 might have duplicate values. When a *CPA* is composed from two *CPPs*, duplicate ID attribute
 3223 values SHALL be tested for. If a duplicate ID attribute value is present, one of the duplicates
 3224 SHALL be given a new value and the corresponding IDREF attribute values from the
 3225 corresponding *CPP* SHALL be corrected.
 3226

3227 NOTE: A party can seek to prevent ID/IDREF reassignment in the *CPA* by choosing ID
 3228 and IDREF values which are likely to be unique among its trading partners. For example,
 3229 the following *Certificate* element found in a *CPP* has a *certId* attribute that is generic
 3230 enough that it might clash with a *certId* attribute found in a collaborating party's *CPP*:

```
3231 <tp:Certificate
3232 tp:certId="EncryptionCert"><ds:KeyInfo/></tp:Certificate>
```

3234 To prevent reassignment of this ID (and its associated IDREFs) in a *CPA*, a better choice
 3235 of *certId* in Company A's *CPP* would be:

```
3236 <tp:Certificate
3237 tp:certId="CompanyA_EncryptionCert"><ds:KeyInfo/></tp:Certificate>
```

3240 9.12 Modifying Parameters of the Process-Specification Document Based on 3241 Information in the CPA

3242 A *Process-Specification* document contains a number of parameters, expressed as XML
 3243 attributes. An example is the security attributes that are counterparts of the attributes of the *CPA*
 3244 *BusinessTransactionCharacteristics* element. The values of these attributes can be considered to
 3245 be default values or recommendations. When a *CPA* is created, the *Parties* might decide to
 3246 accept the recommendations in the *Process-Specification* or they MAY agree on values of these
 3247 parameters that better reflect their needs.
 3248

3249 When a *CPA* is used to configure a run-time system, choices specified in the *CPA* MUST always
 3250 assume precedence over choices specified in the referenced *Process-Specification* document. In
 3251 particular, all choices expressed in a *CPA*'s *BusinessTransactionCharacteristics* and *Packaging*
 3252 elements MUST be implemented as agreed to by the *Parties*. These choices SHALL override
 3253 the default values expressed in the *Process-Specification* document. The process of installing the
 3254 information from the *CPA* and *Process-Specification* document MUST verify that all of the
 3255 resulting choices are mutually consistent and MUST signal an error if they are not.
 3256

3257 NOTE: There are several ways of overriding the information in the *Process-*
 3258 *Specification* document by information from the *CPA*. For example:

- 3259 • The *CPA* composition tool can create a separate copy of the *Process-Specification*

- 3261 document. The tool can then directly modify the *Process-Specification* document
3262 with information from the *CPA*. An advantage of this method is that the override
3263 process is performed entirely by the *CPA* composition tool.
- 3264 • A *CPA* installation tool can dynamically override parameters in the *Process-*
3265 *Specification* document using information from the corresponding parameters in the
3266 *CPA* at the time the *CPA* and *Process-Specification* document are installed in the
3267 *Parties'* systems. This eliminates the need to create a separate copy of the *Process-*
3268 *Specification* document.
 - 3269 • Other possible methods might be based on XSLT transformations of the parameter
3270 information in the *CPA* and/or the *Process-Specification* document.

3271 **10 References**

3272 Some references listed below specify functions for which specific XML definitions are provided
3273 in the *CPP* and *CPA*. Other specifications are referred to in this specification in the sense that
3274 they are represented by keywords for which the *Parties* to the *CPA* MAY obtain plug-ins or
3275 write custom support software but do not require specific XML element sets in the *CPP* and
3276 *CPA*.

3277
3278 In a few cases, the only available specification for a function is a proprietary specification.
3279 These are indicated by notes within the citations below.

3280
3281 [ccOVER] ebXML Core Components Overview, <http://www.ebxml.org/specs/ccOVER.pdf>.

3282
3283 [DIGENV] Digital Envelope, RSA Laboratories, <http://www.rsasecurity.com/rsalabs/faq/2-2-4.html>.
3284 NOTE: At this time, the only available specification for digital envelope appears to be the RSA
3285 Laboratories specification.

3286
3287 [ebBPSS] ebXML Business Process Specification Schema, <http://www.ebxml.org/specs/ebBPSS.pdf>.

3288
3289 [ebMS] ebXML Message Service Specification, [http://www.oasis-open.org/committees/ebxml-
3290 msg/documents/ebMS_v2_0.pdf](http://www.oasis-open.org/committees/ebxml-
3290 msg/documents/ebMS_v2_0.pdf).

3291
3292 [ebRS] ebXML Registry Services Specification, [http://www.oasis-
3293 open.org/committees/regrep/documents/2.0/specs/ebrs.pdf](http://www.oasis-
3293 open.org/committees/regrep/documents/2.0/specs/ebrs.pdf).

3294
3295 [HTTP] Hypertext Transfer Protocol, Internet Engineering Task Force RFC 2616, [http://www.rfc-
3296 editor.org/rfc/rfc2616.txt](http://www.rfc-
3296 editor.org/rfc/rfc2616.txt).

3297
3298 [IPSEC] IP Security Document Roadmap, Internet Engineering Task Force RFC 2411,
3299 <http://www.ietf.org/rfc/rfc2411.txt>.

3300
3301 [ISO6523] Structure for the Identification of Organizations and Organization Parts, International
3302 Standards Organization ISO-6523.

3303
3304 [MIME] MIME (Multipurpose Internet Mail Extensions) Part One: Mechanisms for Specifying
3305 and Describing the Format of Internet *Message* Bodies. Internet Engineering Task Force RFC
3306 1521, <http://www.ietf.org/rfc/rfc1521.txt>.

3307
3308 [RFC959] File Transfer Protocol (FTP), Internet Engineering Task Force RFC 959,
3309 <http://www.ietf.org/rfc/rfc959.txt>.

3310
3311 [RFC1123] Requirements for Internet Hosts -- Application and Support, Internet Engineering
3312 Task Force RFC 1123, <http://www.ietf.org/rfc/rfc1123.txt>.

3313

- 3314 [RFC1579] Firewall-Friendly FTP, Internet Engineering Task Force RFC 1579,
3315 <http://www.ietf.org/rfc/rfc1579.txt>.
3316
- 3317 [RFC2015] MIME Security with Pretty Good Privacy, Internet Engineering Task Force, RFC
3318 2015, <http://www.ietf.org/rfc/rfc2015.txt>.
3319
- 3320 [RFC2119] Key Words for use in RFCs to Indicate Requirement Levels, Internet Engineering
3321 Task Force RFC 2119, <http://www.ietf.org/rfc/rfc2119.txt>.
3322
- 3323 [RFC2246] The TLS Protocol, Internet Engineering Task Force RFC 2246,
3324 <http://www.ietf.org/rfc/rfc2246.txt>.
3325
- 3326 [RFC2251] Lightweight Directory Access Protocol (v3), Internet Engineering Task Force RFC
3327 2251, <http://www.ietf.org/rfc/rfc2251.txt>.
3328
- 3329 [RFC2396] Uniform Resource Identifiers (URI): Generic Syntax, Internet Engineering Task
3330 Force RFC 2396, <http://www.ietf.org/rfc/rfc2396.txt>.
3331
- 3332 [RFC2617] HTTP Authentication: Basic and Digest Authentication, , Internet Engineering Task
3333 Force RFC 2617, <http://www.ietf.org/rfc/rfc2617.txt>.
3334
- 3335 [RFC2822] Internet Message Format, Internet Engineering Task Force RFC 2822,
3336 <http://www.ietf.org/rfc/rfc2822.txt>.
3337
- 3338 [S/MIME] S/MIME Version 3 Message Specification, Internet Engineering Task Force RFC
3339 2633, <http://www.ietf.org/rfc/rfc2633.txt>.
3340
- 3341 [SAML] Security Assertion Markup Language, [http://www.oasis-open.org/committees/security/-
3342 documents](http://www.oasis-open.org/committees/security/-documents).
3343
- 3344 [SMTP] Simple Mail Transfer Protocol, Internet Engineering Task Force RFC 2821,
3345 <http://www.faqs.org/rfcs/rfc2821.html>.
3346
- 3347 [SSL] Secure Sockets Layer, Netscape Communications Corp., <http://www.netscape.com/eng/ssl3/>
3348 NOTE: At this time, it appears that the Netscape specification is the only available specification
3349 of SSL.
3350
- 3351 [X12] ANSI X12 Standard for Electronic Data Interchange, X12 Standard Release
3352 4050, December 2001.
3353
- 3354 [XAML] Transaction Authority Markup Language, <http://xaml.org/>.
3355
- 3356 [XLINK] XML Linking Language, <http://www.w3.org/TR/xlink/>.
3357
- 3358 [XML] Extensible Markup Language (XML), World Wide Web Consortium,

- 3359 <http://www.w3.org/XML>.
- 3360
- 3361 [XMLC14N] Canonical XML, Ver. 1.0, Worldwide Web Consortium,
- 3362 <http://www.w3.org/TR/2001/REC-xml-c14n-20010315>.
- 3363
- 3364 [XMLDSIG] XML Signature Syntax and Processing, Worldwide Web Consortium,
- 3365 <http://www.w3.org/TR/xmlsig-core/>.
- 3366
- 3367 [XMLENC] XML Encryption Syntax and Processing, Worldwide Web Consortium,
- 3368 <http://www.w3.org/TR/2002/CR-xmlenc-core-20020304/>.
- 3369
- 3370 [XMLNS] Namespaces in XML, Worldwide Web Consortium, [http://www.w3.org/TR/REC-xml-](http://www.w3.org/TR/REC-xml-names/)
- 3371 [names/](http://www.w3.org/TR/REC-xml-names/).
- 3372
- 3373 [XMLSCHEMA-1] XML Schema Part 1: Structures, Worldwide Web Consortium,
- 3374 <http://www.w3.org/TR/xmlschema-1/>.
- 3375
- 3376 [XMLSCHEMA-2] XML Schema Part 2: Datatypes, Worldwide Web Consortium,
- 3377 <http://www.w3.org/TR/xmlschema-2/>.
- 3378
- 3379 [XPOINTER] XML Pointer Language, Worldwide Web Consortium, <http://www.w3.org/TR/xptr/>.
- 3380
- 3381 [ZLIB] Zlib: A Massively Spiffy Yet Delicately Unobtrusive Compression Library,
- 3382 <http://www.gzip.org/zlib/>.

3383 **11 Conformance**

3384 In order to conform to this specification, an implementation:

- 3385 a) SHALL support all the functional and interface requirements defined in this specification,
- 3386 b) SHALL NOT specify any requirements that would contradict or cause non-conformance
- 3387 to this specification.

3388

3389 A conforming implementation SHALL satisfy the conformance requirements of the applicable
3390 parts of this specification.

3391

3392 An implementation of a tool or service that creates or maintains ebXML *CPP* or *CPA* instance
3393 documents SHALL be determined to be conformant by validation of the *CPP* or *CPA* instance
3394 documents, created or modified by said tool or service, against the XML
3395 Schema[XMLSCHEMA-1] definition of the *CPP* or *CPA* in Appendix D and available from

3396

3397 http://www.oasis-open.org/committees/ebxml-cppa/schema/cpp-cpa-2_0.xsd

3398

3399 by using two or more validating XML Schema parsers that conform to the W3C XML Schema
3400 specifications[XMLSCHEMA-1, XMLSCHEMA-2].

3401

3402 The objective of conformance testing is to determine whether an implementation being tested
3403 conforms to the requirements stated in this specification. Conformance testing enables vendors to
3404 implement compatible and interoperable systems. Implementations and applications SHALL be
3405 tested using available test suites to verify their conformance to this specification.

3406

3407 Publicly available test suites from vendor neutral organizations such as OASIS and the U.S.A.
3408 National Institute of Science and Technology (NIST) SHOULD be used to verify the
3409 conformance of implementations, applications, and components claiming conformance to this
3410 specification. Open-source reference implementations might be available to allow vendors to test
3411 their products for interface compatibility, conformance, and interoperability.

3412

3413 12 Disclaimer

3414 The views and specification expressed in this document are those of the authors and are not
3415 necessarily those of their employers. The authors and their employers specifically disclaim
3416 responsibility for any problems arising from correct or incorrect implementation or use of this
3417 design.

3418 13 Contact Information

3419

3420 Arvola Chan (Author)

3421 TIBCO Software

3422 3303 Hillview Avenue

3423 Palo Alto, CA 94304

3424 USA

3425 Phone: 650-846-5046

3426 email: <mailto:arvola@tibco.com>

3427

3428 Dale W. Moberg (Author)

3429 Cyclone Commerce

3430 8388 E. Hartford Drive

3431 Scottsdale, AZ 85255

3432 USA

3433 Phone: 480-627-2648

3434 email: <mailto:dmoberg@cyclonecommerce.com>

3435

3436 Himagiri Mukkamala (Author)

3437 Sybase Inc.

3438 5000 Hacienda Dr

3439 Dublin, CA, 84568

3440 USA

3441 Phone: 925-236-5477

3442 email: <mailto:himagiri@sybase.com>

3443

3444 Peter M. Ogden (Author)

3445 Cyclone Commerce, Inc.

3446 8388 East Hartford Drive

3447 Scottsdale, AZ 85255

3448 USA

3449 Phone: 480-627-1800

3450 email: <mailto:pogden@cyclonecommerce.com>

3451

3452 Martin W. Sachs (Author)

3453 IBM T. J. Watson Research Center

3454 P.O.B. 704

3455 Yorktown Hts, NY 10598

3456 USA

3457 Phone: 914-784-7287

3458 email: <mailto:mwsachs@us.ibm.com>

3459

3460 Tony Weida (Coordinating Editor)

3461 535 West 110th St., #4J
3462 New York, NY 10025
3463 USA
3464 Phone: 212-678-5265
3465 email: <mailto:rweida@hotmail.com>
3466
3467 Jean Zheng
3468 Vitria
3469 945 Stewart Drive
3470 Sunnyvale, CA 94086
3471 USA
3472 Phone: 408-212-2468
3473 email: <mailto:jzheng@vitria.com>

3474 **Notices**

3475

3476 Portions of this document are copyright (c) 2001 OASIS and UN/CEFACT.

3477

3478 **Copyright (C) The Organization for the Advancement of Structured Information**
3479 **Standards [OASIS] 2002. All Rights Reserved.**

3480

3481 This document and translations of it may be copied and furnished to others, and derivative works
3482 that comment on or otherwise explain it or assist in its implementation may be prepared, copied,
3483 published and distributed, in whole or in part, without restriction of any kind, provided that the
3484 above copyright notice and this paragraph are included on all such copies and derivative works.
3485 However, this document itself may not be modified in any way, such as by removing the
3486 copyright notice or references to OASIS, except as needed for the purpose of developing OASIS
3487 specifications, in which case the procedures for copyrights defined in the OASIS Intellectual
3488 Property Rights document must be followed, or as required to translate it into languages other
3489 than English.

3490

3491 The limited permissions granted above are perpetual and will not be revoked by OASIS or its
3492 successors or assigns.

3493

3494 This document and the information contained herein is provided on an "AS IS" basis and OASIS
3495 DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT
3496 LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN
3497 WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF
3498 MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

3499

3500 OASIS takes no position regarding the validity or scope of any intellectual property or other
3501 rights that might be claimed to pertain to the implementation or use of the technology described
3502 in this document or the extent to which any license under such rights might or might not be
3503 available; neither does it represent that it has made any effort to identify any such rights.
3504 Information on OASIS's procedures with respect to rights in OASIS specifications can be found
3505 at the OASIS website. Copies of claims of rights made available for publication and any
3506 assurances of licenses to be made available, or the result of an attempt made to obtain a general
3507 license or permission for the use of such proprietary rights by implementers or users of this
3508 specification, can be obtained from the OASIS Executive Director.

3509

3510 OASIS invites any interested party to bring to its attention any copyrights, patents or patent
3511 applications, or other proprietary rights which may cover technology that may be required to
3512 implement this specification. Please address the information to the OASIS Executive Director.

3513

3514 OASIS has been notified of intellectual property rights claimed in regard to some or all of the
3515 contents of this specification. For more information consult the online list of claimed rights.

3516 **Appendix A Example of CPP Document (Non-Normative)**

3517 This example includes two CPPs that are used to form the CPA in Appendix B. They are
3518 available as ASCII files at

3519 [http://www.oasis-open.org/committees/ebxml-cppa/schema/cpp-example-companyA-
3520 2_0b.xml](http://www.oasis-open.org/committees/ebxml-cppa/schema/cpp-example-companyA-2_0b.xml)

3521 [http://www.oasis-open.org/committees/ebxml-cppa/schema/cpp-example-companyB-
3522 2_0b.xml](http://www.oasis-open.org/committees/ebxml-cppa/schema/cpp-example-companyB-2_0b.xml)

```
3523
3524 cpp-example-companyA-2_0b.xml:
3525
3526 <?xml version="1.0"?>
3527 <!-- Copyright UN/CEFACT and OASIS, 2001. All Rights Reserved. -->
3528 <tp:CollaborationProtocolProfile
3529   xmlns:tp="http://www.oasis-open.org/committees/ebxml-cppa/schema/cpp-cpa-2_0.xsd"
3530   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3531   xmlns:xlink="http://www.w3.org/1999/xlink"
3532   xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
3533   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
3534   xsi:schemaLocation="http://www.oasis-open.org/committees/ebxml-cppa/schema/cpp-cpa-2_0.xsd
3535     cpp-cpa-2_0.xsd"
3536   tp:cppid="uri:companyA-cpp" tp:version="2_0b">
3537 <!-- Party info for CompanyA-->
3538 <tp:PartyInfo
3539   tp:partyName="CompanyA"
3540   tp:defaultMshChannelId="asyncChannelA1"
3541   tp:defaultMshPackageId="CompanyA_MshSignalPackage">
3542 <tp:PartyId
3543   tp:type="urn:oasis:names:tc:ebxml-cppa:partyid-type:duns">123456789</tp:PartyId>
3544 <tp:PartyRef xlink:href="http://CompanyA.com/about.html"/>
3545 <tp:CollaborationRole>
3546 <tp:ProcessSpecification
3547   tp:version="2.0"
3548   tp:name="PIP3A4RequestPurchaseOrder"
3549   xlink:type="simple"
3550   xlink:href="http://www.rosettanet.org/processes/3A4.xml"
3551   tp:uuid="urn:icann:rosettanet.org:bpid:3A4$2.0"/>
3552 <tp:Role
3553   tp:name="Buyer"
3554   xlink:type="simple"
3555   xlink:href="http://www.rosettanet.org/processes/3A4.xml#Buyer"/>
3556 <tp:ApplicationCertificateRef tp:certId="CompanyA_AppCert"/>
3557 <tp:ServiceBinding>
3558 <tp:Service>bpid:icann:rosettanet.org:3A4$2.0</tp:Service>
3559 <tp:CanSend>
3560 <tp:ThisPartyActionBinding
3561   tp:id="companyA_ABID1"
3562   tp:action="Purchase Order Request Action"
3563   tp:packageId="CompanyA_RequestPackage">
3564 <tp:BusinessTransactionCharacteristics
3565   tp:isNonRepudiationRequired="true"
3566   tp:isNonRepudiationReceiptRequired="true"
3567   tp:isConfidential="transient"
3568   tp:isAuthenticated="persistent"
3569   tp:isTamperProof="persistent"
3570   tp:isAuthorizationRequired="true"
3571   tp:timeToAcknowledgeReceipt="PT2H"
3572   tp:timeToPerform="P1D"/>
3573 <tp>ActionContext
3574   tp:binaryCollaboration="Request Purchase Order"
3575   tp:businessTransactionActivity="Request Purchase Order"
3576   tp:requestOrResponseAction="Purchase Order Request Action"/>
3577 <tp:ChannelId>asyncChannelA1</tp:ChannelId>
3578 </tp:ThisPartyActionBinding>
3579 </tp:CanSend>
3580 <tp:CanSend>
3581 <tp:ThisPartyActionBinding
3582   tp:id="companyA_ABID2"
3583   tp:action="ReceiptAcknowledgement"
3584   tp:packageId="CompanyA_ReceiptAcknowledgmentPackage">
```



```

3585      <tp:BusinessTransactionCharacteristics
3586        tp:isNonRepudiationRequired="true"
3587        tp:isNonRepudiationReceiptRequired="true"
3588          tp:isConfidential="transient"
3589        tp:isAuthenticated="persistent"
3590        tp:isTamperProof="persistent"
3591        tp:isAuthorizationRequired="true"/>
3592      <tp:ChannelId>asyncChannelA1</tp:ChannelId>
3593    </tp:ThisPartyActionBinding>
3594  </tp:CanSend>
3595  <!-- The next binding uses a synchronous delivery channel -->
3596  <tp:CanSend>
3597    <tp:ThisPartyActionBinding
3598      tp:id="companyA_ABID6"
3599      tp:action="Purchase Order Request Action"
3600      tp:packageId="CompanyA_RequestPackage">
3601      <tp:BusinessTransactionCharacteristics
3602        tp:isNonRepudiationRequired="true"
3603        tp:isNonRepudiationReceiptRequired="true"
3604          tp:isConfidential="transient"
3605        tp:isAuthenticated="persistent"
3606        tp:isTamperProof="persistent"
3607        tp:isAuthorizationRequired="true"
3608        tp:timeToAcknowledgeReceipt="PT5M"
3609        tp:timeToPerform="PT5M"/>
3610      <tp:ActionContext
3611        tp:binaryCollaboration="Request Purchase Order"
3612        tp:businessTransactionActivity="Request Purchase Order"
3613        tp:requestOrResponseAction="Purchase Order Request Action"/>
3614      <tp:ChannelId>syncChannelA1</tp:ChannelId>
3615    </tp:ThisPartyActionBinding>
3616  <tp:CanReceive>
3617    <tp:ThisPartyActionBinding
3618      tp:id="companyA_ABID7"
3619      tp:action="Purchase Order Confirmation Action"
3620      tp:packageId="CompanyA_SyncReplyPackage">
3621      <tp:BusinessTransactionCharacteristics
3622        tp:isNonRepudiationRequired="true"
3623        tp:isNonRepudiationReceiptRequired="true"
3624        tp:isConfidential="transient"
3625        tp:isAuthenticated="persistent"
3626        tp:isTamperProof="persistent"
3627        tp:isAuthorizationRequired="true"
3628        tp:timeToAcknowledgeReceipt="PT5M"/>
3629      <tp:ActionContext
3630        tp:binaryCollaboration="Request Purchase Order"
3631        tp:businessTransactionActivity="Request Purchase Order"
3632        tp:requestOrResponseAction="Purchase Order Confirmation Action"/>
3633      <tp:ChannelId>syncChannelA1</tp:ChannelId>
3634    </tp:ThisPartyActionBinding>
3635  </tp:CanReceive>
3636  <tp:CanReceive>
3637    <tp:ThisPartyActionBinding
3638      tp:id="companyA_ABID8"
3639      tp:action="Exception"
3640      tp:packageId="CompanyA_ExceptionPackage">
3641      <tp:BusinessTransactionCharacteristics
3642        tp:isNonRepudiationRequired="true"
3643        tp:isNonRepudiationReceiptRequired="true"
3644        tp:isConfidential="transient"
3645        tp:isAuthenticated="persistent"
3646        tp:isTamperProof="persistent"
3647        tp:isAuthorizationRequired="true"/>
3648      <tp:ChannelId>syncChannelA1</tp:ChannelId>
3649    </tp:ThisPartyActionBinding>
3650  </tp:CanReceive>
3651  </tp:CanSend>
3652  <tp:CanReceive>
3653    <tp:ThisPartyActionBinding
3654      tp:id="companyA_ABID3"
3655      tp:action="Purchase Order Confirmation Action"
3656      tp:packageId="CompanyA_ResponsePackage">
3657      <tp:BusinessTransactionCharacteristics
3658        tp:isNonRepudiationRequired="true"
3659        tp:isNonRepudiationReceiptRequired="true"
3660        tp:isConfidential="transient"
3661        tp:isAuthenticated="persistent"
3662        tp:isTamperProof="persistent"

```

```

3663         tp:isAuthorizationRequired="true"
3664         tp:timeToAcknowledgeReceipt="PT2H" />
3665     <tp:ActionContext
3666         tp:binaryCollaboration="Request Purchase Order"
3667         tp:businessTransactionActivity="Request Purchase Order"
3668         tp:requestOrResponseAction="Purchase Order Confirmation Action" />
3669     <tp:ChannelId>asyncChannelA1</tp:ChannelId>
3670 </tp:ThisPartyActionBinding>
3671 </tp:CanReceive>
3672 <tp:CanReceive>
3673     <tp:ThisPartyActionBinding
3674         tp:id="companyA_ABID4"
3675         tp:action="ReceiptAcknowledgment"
3676         tp:packageId="CompanyA_ReceiptAcknowledgmentPackage">
3677         <tp:BusinessTransactionCharacteristics
3678             tp:isNonRepudiationRequired="true"
3679             tp:isNonRepudiationReceiptRequired="true"
3680             tp:isConfidential="transient"
3681             tp:isAuthenticated="persistent" tp:isTamperProof="persistent"
3682             tp:isAuthorizationRequired="true" />
3683         <tp:ChannelId>asyncChannelA1</tp:ChannelId>
3684     </tp:ThisPartyActionBinding>
3685 </tp:CanReceive>
3686 <tp:CanReceive>
3687     <tp:ThisPartyActionBinding
3688         tp:id="companyA_ABID5"
3689         tp:action="Exception"
3690         tp:packageId="CompanyA_ExceptionPackage">
3691         <tp:BusinessTransactionCharacteristics
3692             tp:isNonRepudiationRequired="true"
3693             tp:isNonRepudiationReceiptRequired="true"
3694             tp:isConfidential="transient"
3695             tp:isAuthenticated="persistent"
3696             tp:isTamperProof="persistent"
3697             tp:isAuthorizationRequired="true" />
3698         <tp:ChannelId>asyncChannelA1</tp:ChannelId>
3699     </tp:ThisPartyActionBinding>
3700 </tp:CanReceive>
3701 </tp:ServiceBinding>
3702 </tp:CollaborationRole>
3703 <!-- Certificates used by the "Buyer" company -->
3704 <tp:Certificate tp:certId="CompanyA_AppCert">
3705     <ds:KeyInfo>
3706         <ds:KeyName>CompanyA_AppCert_Key</ds:KeyName>
3707     </ds:KeyInfo>
3708 </tp:Certificate>
3709 <tp:Certificate tp:certId="CompanyA_SigningCert">
3710     <ds:KeyInfo>
3711         <ds:KeyName>CompanyA_SigningCert_Key</ds:KeyName>
3712     </ds:KeyInfo>
3713 </tp:Certificate>
3714 <tp:Certificate tp:certId="CompanyA_EncryptionCert">
3715     <ds:KeyInfo>
3716         <ds:KeyName>CompanyA_EncryptionCert_Key</ds:KeyName>
3717     </ds:KeyInfo>
3718 </tp:Certificate>
3719 <tp:Certificate tp:certId="CompanyA_ServerCert">
3720     <ds:KeyInfo>
3721         <ds:KeyName>CompanyA_ServerCert_Key</ds:KeyName>
3722     </ds:KeyInfo>
3723 </tp:Certificate>
3724 <tp:Certificate tp:certId="CompanyA_ClientCert">
3725     <ds:KeyInfo>
3726         <ds:KeyName>CompanyA_ClientCert_Key</ds:KeyName>
3727     </ds:KeyInfo>
3728 </tp:Certificate>
3729 <tp:Certificate tp:certId="TrustedRootCertA1">
3730     <ds:KeyInfo>
3731         <ds:KeyName>TrustedRootCertA1_Key</ds:KeyName>
3732     </ds:KeyInfo>
3733 </tp:Certificate>
3734 <tp:Certificate tp:certId="TrustedRootCertA2">
3735     <ds:KeyInfo>
3736         <ds:KeyName>TrustedRootCertA2_Key</ds:KeyName>
3737     </ds:KeyInfo>
3738 </tp:Certificate>
3739 <tp:Certificate tp:certId="TrustedRootCertA3">
3740     <ds:KeyInfo>

```

```

3741     <ds:KeyName>TrustedRootCertA3_Key</ds:KeyName>
3742     </ds:KeyInfo>
3743 </tp:Certificate>
3744 <tp:Certificate tp:certId="TrustedRootCertA4">
3745     <ds:KeyInfo>
3746         <ds:KeyName>TrustedRootCertA4_Key</ds:KeyName>
3747     </ds:KeyInfo>
3748 </tp:Certificate>
3749 <tp:Certificate tp:certId="TrustedRootCertA5">
3750     <ds:KeyInfo>
3751         <ds:KeyName>TrustedRootCertA5_Key</ds:KeyName>
3752     </ds:KeyInfo>
3753 </tp:Certificate>
3754 <tp:SecurityDetails tp:securityId="CompanyA_TransportSecurity">
3755     <tp:TrustAnchors>
3756         <tp:AnchorCertificateRef tp:certId="TrustedRootCertA1"/>
3757         <tp:AnchorCertificateRef tp:certId="TrustedRootCertA2"/>
3758         <tp:AnchorCertificateRef tp:certId="TrustedRootCertA4"/>
3759     </tp:TrustAnchors>
3760 </tp:SecurityDetails>
3761 <tp:SecurityDetails tp:securityId="CompanyA_MessageSecurity">
3762     <tp:TrustAnchors>
3763         <tp:AnchorCertificateRef tp:certId="TrustedRootCertA3"/>
3764         <tp:AnchorCertificateRef tp:certId="TrustedRootCertA5"/>
3765     </tp:TrustAnchors>
3766 </tp:SecurityDetails>
3767 <!-- An asynchronous delivery channel -->
3768 <tp:DeliveryChannel
3769     tp:channelId="asyncChannelA1"
3770     tp:transportId="transportA2"
3771     tp:docExchangeId="docExchangeA1">
3772     <tp:MessagingCharacteristics
3773         tp:syncReplyMode="none"
3774         tp:ackRequested="always"
3775         tp:ackSignatureRequested="always"
3776         tp:duplicateElimination="always"/>
3777 </tp:DeliveryChannel>
3778 <!-- A synchronous delivery channel -->
3779 <tp:DeliveryChannel
3780     tp:channelId="syncChannelA1"
3781     tp:transportId="transportA1"
3782     tp:docExchangeId="docExchangeA1">
3783     <tp:MessagingCharacteristics
3784         tp:syncReplyMode="signalsAndResponse"
3785         tp:ackRequested="always"
3786         tp:ackSignatureRequested="always"
3787         tp:duplicateElimination="always"/>
3788 </tp:DeliveryChannel>
3789 <tp:Transport tp:transportId="transportA1">
3790     <tp:TransportSender>
3791         <tp:TransportProtocol tp:version="1.1">HTTP</tp:TransportProtocol>
3792         <tp:AccessAuthentication>basic</tp:AccessAuthentication>
3793         <tp:AccessAuthentication>digest</tp:AccessAuthentication>
3794         <tp:TransportClientSecurity>
3795             <tp:TransportSecurityProtocol tp:version="3.0">SSL</tp:TransportSecurityProtocol>
3796             <tp:ClientCertificateRef tp:certId="CompanyA_ClientCert"/>
3797             <tp:ServerSecurityDetailsRef tp:securityId="CompanyA_TransportSecurity"/>
3798         </tp:TransportClientSecurity>
3799     </tp:TransportSender>
3800     <tp:TransportReceiver>
3801         <tp:TransportProtocol tp:version="1.1">HTTP</tp:TransportProtocol>
3802         <tp:AccessAuthentication>basic</tp:AccessAuthentication>
3803         <tp:AccessAuthentication>digest</tp:AccessAuthentication>
3804         <tp:Endpoint
3805             tp:uri="https://www.CompanyA.com/servlets/ebxmlhandler/sync"
3806             tp:type="allPurpose"/>
3807         <tp:TransportServerSecurity>
3808             <tp:TransportSecurityProtocol tp:version="3.0">SSL</tp:TransportSecurityProtocol>
3809             <tp:ServerCertificateRef tp:certId="CompanyA_ServerCert"/>
3810             <tp:ClientSecurityDetailsRef tp:securityId="CompanyA_TransportSecurity"/>
3811         </tp:TransportServerSecurity>
3812     </tp:TransportReceiver>
3813 </tp:Transport>
3814 <tp:Transport tp:transportId="transportA2">
3815     <tp:TransportSender>
3816         <tp:TransportProtocol tp:version="1.1">HTTP</tp:TransportProtocol>
3817         <tp:AccessAuthentication>basic</tp:AccessAuthentication>
3818         <tp:AccessAuthentication>digest</tp:AccessAuthentication>

```

```

3819
3920
4021
4122
4223
4324
4425
4526
4627
4728
4829
4930
5031
5132
5233
5334
5435
5536
5637
5738
5839
5940
6041
6142
6243
6344
6445
6546
6647
6748
6849
6950
7051
7152
7253
7354
7455
7556
7657
7758
7859
7960
8061
8162
8263
8364
8465
8566
8667
8768
8869
8970
9071
9172
9273
9374
9475
9576
9677
9778
9879
9980

```

```

    <tp:TransportClientSecurity>
      <tp:TransportSecurityProtocol tp:version="3.0">SSL</tp:TransportSecurityProtocol>
      <tp:ClientCertificateRef tp:certId="CompanyA_ClientCert"/>
      <tp:ServerSecurityDetailsRef tp:securityId="CompanyA_TransportSecurity"/>
    </tp:TransportClientSecurity>
  </tp:TransportSender>
  <tp:TransportReceiver>
    <tp:TransportProtocol tp:version="1.1">HTTP</tp:TransportProtocol>
    <tp:AccessAuthentication>basic</tp:AccessAuthentication>
    <tp:AccessAuthentication>digest</tp:AccessAuthentication>
    <tp:Endpoint
      tp:uri="https://www.CompanyA.com/servlets/ebxmlhandler/sync"
      tp:type="allPurpose"/>
    <tp:TransportServerSecurity>
      <tp:TransportSecurityProtocol tp:version="3.0">SSL</tp:TransportSecurityProtocol>
      <tp:ServerCertificateRef tp:certId="CompanyA_ServerCert"/>
      <tp:ClientSecurityDetailsRef tp:securityId="CompanyA_TransportSecurity"/>
    </tp:TransportServerSecurity>
  </tp:TransportReceiver>
</tp:Transport>
<tp:DocExchange tp:docExchangeId="docExchangeA1">
  <tp:ebXMLSenderBinding tp:version="2.0">
    <tp:ReliableMessaging>
      <tp:Retries>3</tp:Retries>
      <tp:RetryInterval>PT2H</tp:RetryInterval>
      <tp:MessageOrderSemantics>Guaranteed</tp:MessageOrderSemantics>
    </tp:ReliableMessaging>
    <tp:PersistDuration>P1D</tp:PersistDuration>
    <tp:SenderNonRepudiation>
      <tp:NonRepudiationProtocol>http://www.w3.org/2000/09/xmldsig#</tp:NonRepudiationProtocol>
      <tp:HashFunction>http://www.w3.org/2000/09/xmldsig#sha1</tp:HashFunction>
      <tp:SignatureAlgorithm>http://www.w3.org/2000/09/xmldsig#dsa-
sha1</tp:SignatureAlgorithm>
      <tp:SigningCertificateRef tp:certId="CompanyA_SigningCert"/>
    </tp:SenderNonRepudiation>
    <tp:SenderDigitalEnvelope>
      <tp:DigitalEnvelopeProtocol tp:version="2.0">S/MIME</tp:DigitalEnvelopeProtocol>
      <tp:EncryptionAlgorithm>DES-CBC</tp:EncryptionAlgorithm>
      <tp:EncryptionSecurityDetailsRef tp:securityId="CompanyA_MessageSecurity"/>
    </tp:SenderDigitalEnvelope>
  </tp:ebXMLSenderBinding>
  <tp:ebXMLReceiverBinding tp:version="2.0">
    <tp:ReliableMessaging>
      <tp:Retries>3</tp:Retries>
      <tp:RetryInterval>PT2H</tp:RetryInterval>
      <tp:MessageOrderSemantics>Guaranteed</tp:MessageOrderSemantics>
    </tp:ReliableMessaging>
    <tp:PersistDuration>P1D</tp:PersistDuration>
    <tp:ReceiverNonRepudiation>
      <tp:NonRepudiationProtocol>http://www.w3.org/2000/09/xmldsig#</tp:NonRepudiationProtocol>
      <tp:HashFunction>http://www.w3.org/2000/09/xmldsig#sha1</tp:HashFunction>
      <tp:SignatureAlgorithm>http://www.w3.org/2000/09/xmldsig#dsa-
sha1</tp:SignatureAlgorithm>
      <tp:SigningSecurityDetailsRef tp:securityId="CompanyA_MessageSecurity"/>
    </tp:ReceiverNonRepudiation>
    <tp:ReceiverDigitalEnvelope>
      <tp:DigitalEnvelopeProtocol tp:version="2.0">S/MIME</tp:DigitalEnvelopeProtocol>
      <tp:EncryptionAlgorithm>DES-CBC</tp:EncryptionAlgorithm>
      <tp:EncryptionCertificateRef tp:certId="CompanyA_EncryptionCert"/>
    </tp:ReceiverDigitalEnvelope>
  </tp:ebXMLReceiverBinding>
</tp:DocExchange>
</tp:PartyInfo>
<!-- SimplePart corresponding to the SOAP Envelope -->
<tp:SimplePart
  tp:id="CompanyA_MsgHdr"
  tp:mimetype="text/xml">
  <tp:NamespaceSupported
    tp:location="http://www.oasis-open.org/committees/ebxml-msg/schema/msg-header-2_0.xsd"
    tp:version="2.0">
    http://www.oasis-open.org/committees/ebxml-msg/schema/msg-header-2_0.xsd
  </tp:NamespaceSupported>
</tp:SimplePart>
<!-- SimplePart corresponding to a Receipt Acknowledgment business signal -->
<tp:SimplePart
  tp:id="CompanyA_ReceiptAcknowledgment"

```

```

3897 tp:mimetype="application/xml">
3898 <tp:NamespaceSupported
3899   tp:location="http://www.ebxml.org/bpss/ReceiptAcknowledgment.xsd"
3900   tp:version="2.0">
3901   http://www.ebxml.org/bpss/ReceiptAcknowledgment.xsd
3902 </tp:NamespaceSupported>
3903 </tp:SimplePart>
3904 <!-- SimplePart corresponding to an Exception business signal -->
3905 <tp:SimplePart
3906   tp:id="CompanyA_Exception"
3907   tp:mimetype="application/xml">
3908   <tp:NamespaceSupported
3909     tp:location="http://www.oasis-open.org/committees/ebxml-msg/schema/msg-header-2_0.xsd"
3910     tp:version="2.0">
3911     http://www.oasis-open.org/committees/ebxml-msg/schema/msg-header-2_0.xsd
3912   </tp:NamespaceSupported>
3913 </tp:SimplePart>
3914 <!-- SimplePart corresponding to a request action -->
3915 <tp:SimplePart
3916   tp:id="CompanyA_Request"
3917   tp:mimetype="application/xml">
3918   <tp:NamespaceSupported
3919     tp:location="http://www.rosettanet.org/schemas/PIP3A4RequestPurchaseOrder.xsd"
3920     tp:version="2.0">
3921     http://www.rosettanet.org/schemas/PIP3A4RequestPurchaseOrder.xsd
3922   </tp:NamespaceSupported>
3923 </tp:SimplePart>
3924 <!-- SimplePart corresponding to a response action -->
3925 <tp:SimplePart
3926   tp:id="CompanyA_Response"
3927   tp:mimetype="application/xml">
3928   <tp:NamespaceSupported
3929     tp:location="http://www.rosettanet.org/schemas/PurchaseOrderConfirmation.xsd"
3930     tp:version="2.0">
3931     http://www.rosettanet.org/schemas/PIP3A4PurchaseOrderConfirmation.xsd
3932   </tp:NamespaceSupported>
3933 </tp:SimplePart>
3934 <!-- An ebXML message with a SOAP Envelope only -->
3935 <tp:Packaging tp:id="CompanyA_MshSignalPackage">
3936   <tp:ProcessingCapabilities
3937     tp:parse="true"
3938     tp:generate="true"/>
3939   <tp:CompositeList>
3940     <tp:Composite
3941       tp:id="CompanyA_MshSignal"
3942       tp:mimetype="multipart/related"
3943       tp:mimeparameters="type=text/xml">
3944       <tp:Constituent tp:idref="CompanyA_MsgHdr"/>
3945     </tp:Composite>
3946   </tp:CompositeList>
3947 </tp:Packaging>
3948 <!-- An ebXML message with a SOAP Envelope plus a request action payload -->
3949 <tp:Packaging tp:id="CompanyA_RequestPackage">
3950   <tp:ProcessingCapabilities
3951     tp:parse="true"
3952     tp:generate="true"/>
3953   <tp:CompositeList>
3954     <tp:Composite
3955       tp:id="CompanyA_RequestMsg"
3956       tp:mimetype="multipart/related"
3957       tp:mimeparameters="type=text/xml">
3958       <tp:Constituent tp:idref="CompanyA_MsgHdr"/>
3959       <tp:Constituent tp:idref="CompanyA_Request"/>
3960     </tp:Composite>
3961   </tp:CompositeList>
3962 </tp:Packaging>
3963 <!-- An ebXML message with a SOAP Envelope plus a response action payload -->
3964 <tp:Packaging tp:id="CompanyA_ResponsePackage">
3965   <tp:ProcessingCapabilities
3966     tp:parse="true"
3967     tp:generate="true"/>
3968   <tp:CompositeList>
3969     <tp:Composite
3970       tp:id="CompanyA_ResponseMsg"
3971       tp:mimetype="multipart/related"
3972       tp:mimeparameters="type=text/xml">
3973       <tp:Constituent tp:idref="CompanyA_MsgHdr"/>
3974       <tp:Constituent tp:idref="CompanyA_Response"/>

```

```

3975     </tp:Composite>
3976   </tp:CompositeList>
3977 </tp:Packaging>
3978 <!-- An ebXML message with a Receipt Acknowledgment signal, plus a business response,
3979       or an ebXML message with an Exception signal -->
3980 <tp:Packaging tp:id="CompanyA_SyncReplyPackage">
3981   <tp:ProcessingCapabilities
3982     tp:parse="true"
3983     tp:generate="true"/>
3984   <tp:CompositeList>
3985     <tp:Composite
3986       tp:id="CompanyA_SignalAndResponseMsg"
3987       tp:mimetype="multipart/related"
3988       tp:mimeparameters="type=text/xml">
3989         <tp:Constituent tp:idref="CompanyA_MsgHdr"/>
3990         <tp:Constituent tp:idref="CompanyA_ReceiptAcknowledgment"/>
3991         <tp:Constituent tp:idref="CompanyA_Response"/>
3992       </tp:Composite>
3993     </tp:CompositeList>
3994 </tp:Packaging>
3995 <!-- An ebXML message with a SOAP Envelope plus a ReceiptAcknowledgment payload -->
3996 <tp:Packaging tp:id="CompanyA_ReceiptAcknowledgmentPackage">
3997   <tp:ProcessingCapabilities
3998     tp:parse="true"
3999     tp:generate="true"/>
4000   <tp:CompositeList>
4001     <tp:Composite
4002       tp:id="CompanyA_ReceiptAcknowledgmentMsg"
4003       tp:mimetype="multipart/related"
4004       tp:mimeparameters="type=text/xml">
4005         <tp:Constituent tp:idref="CompanyA_MsgHdr"/>
4006         <tp:Constituent tp:idref="CompanyA_ReceiptAcknowledgment"/>
4007       </tp:Composite>
4008     </tp:CompositeList>
4009 </tp:Packaging>
4010 <!-- An ebXML message with a SOAP Envelope plus an Exception payload -->
4011 <tp:Packaging tp:id="CompanyA_ExceptionPackage">
4012   <tp:ProcessingCapabilities
4013     tp:parse="true"
4014     tp:generate="true"/>
4015   <tp:CompositeList>
4016     <tp:Composite
4017       tp:id="CompanyA_ExceptionMsg"
4018       tp:mimetype="multipart/related"
4019       tp:mimeparameters="type=text/xml">
4020         <tp:Constituent tp:idref="CompanyA_MsgHdr"/>
4021         <tp:Constituent tp:idref="CompanyA_Exception"/>
4022       </tp:Composite>
4023     </tp:CompositeList>
4024 </tp:Packaging>
4025 <tp:Comment xml:lang="en-US">Buyer's Collaboration Protocol Profile</tp:Comment>
4026 </tp:CollaborationProtocolProfile>
4027
4028
4029 cpp-example-companyB-2_0b.xml:
4030
4031 <?xml version="1.0"?>
4032 <!-- Copyright UN/CEFACT and OASIS, 2001. All Rights Reserved. -->
4033 <tp:CollaborationProtocolProfile
4034   xmlns:tp="http://www.oasis-open.org/committees/ebxml-cppa/schema/cpp-cpa-2_0.xsd"
4035   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4036   xmlns:xlink="http://www.w3.org/1999/xlink"
4037   xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
4038   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
4039   xsi:schemaLocation="http://www.oasis-open.org/committees/ebxml-cppa/schema/cpp-cpa-2_0.xsd
4040                       cpp-cpa-2_0.xsd"
4041   tp:cppid="uri:companyB-cpp"
4042   tp:version="2_0b">
4043 <!-- Party info for CompanyB-->
4044 <tp:PartyInfo
4045   tp:partyName="CompanyB"
4046   tp:defaultMshChannelId="asyncChannelB1"
4047   tp:defaultMshPackageId="CompanyB_MshSignalPackage">
4048 <tp:PartyId tp:type="urn:oasis:names:tc:ebxml-cppa:partyid-type:duns">987654321</tp:PartyId>
4049 <tp:PartyRef xlink:type="simple" xlink:href="http://CompanyB.com/about.html"/>
4050 <tp:CollaborationRole>
4051 <tp:ProcessSpecification
4052   tp:version="2.0"

```

```

4053     tp:name="PIP3A4RequestPurchaseOrder"
4054     xlink:type="simple" xlink:href="http://www.rosettanet.org/processes/3A4.xml"
4055     tp:uuid="urn:icann:rosettanet.org:bpid:3A4$2.0"/>
4056 <tp:Role
4057   tp:name="Seller"
4058   xlink:type="simple"
4059   xlink:href="http://www.rosettanet.org/processes/3A4.xml#seller"/>
4060 <tp:ApplicationCertificateRef tp:certId="CompanyB_AppCert"/>
4061 <tp:ServiceBinding>
4062   <tp:Service>bpid:icann:rosettanet.org:3A4$2.0</tp:Service>
4063   <tp:CanSend>
4064     <tp:ThisPartyActionBinding
4065       tp:id="companyB_ABID1"
4066       tp:action="Purchase Order Confirmation Action"
4067       tp:packageId="CompanyB_ResponsePackage">
4068       <tp:BusinessTransactionCharacteristics
4069         tp:isNonRepudiationRequired="true"
4070         tp:isNonRepudiationReceiptRequired="true"
4071         tp:isConfidential="transient"
4072         tp:isAuthenticated="persistent"
4073         tp:isTamperProof="persistent"
4074         tp:isAuthorizationRequired="true"
4075         tp:timeToAcknowledgeReceipt="PT2H"/>
4076       <tp:ActionContext
4077         tp:binaryCollaboration="Request Purchase Order"
4078         tp:businessTransactionActivity="Request Purchase Order"
4079         tp:requestOrResponseAction="Purchase Order Confirmation Action"/>
4080       <tp:ChannelId>asyncChannelB1</tp:ChannelId>
4081     </tp:ThisPartyActionBinding>
4082   </tp:CanSend>
4083   <tp:CanSend>
4084     <tp:ThisPartyActionBinding
4085       tp:id="companyB_ABID2"
4086       tp:action="ReceiptAcknowledgement"
4087       tp:packageId="CompanyB_ReceiptAcknowledgmentPackage">
4088       <tp:BusinessTransactionCharacteristics
4089         tp:isNonRepudiationRequired="true"
4090         tp:isNonRepudiationReceiptRequired="true"
4091         tp:isConfidential="transient"
4092         tp:isAuthenticated="persistent"
4093         tp:isTamperProof="persistent"
4094         tp:isAuthorizationRequired="true"/>
4095       <tp:ChannelId>asyncChannelB1</tp:ChannelId>
4096     </tp:ThisPartyActionBinding>
4097   </tp:CanSend>
4098   <tp:CanSend>
4099     <tp:ThisPartyActionBinding
4100       tp:id="companyB_ABID3"
4101       tp:action="Exception"
4102       tp:packageId="CompanyB_ExceptionPackage">
4103       <tp:BusinessTransactionCharacteristics
4104         tp:isNonRepudiationRequired="true"
4105         tp:isNonRepudiationReceiptRequired="true"
4106         tp:isConfidential="transient"
4107         tp:isAuthenticated="persistent"
4108         tp:isTamperProof="persistent"
4109         tp:isAuthorizationRequired="true"/>
4110       <tp:ChannelId>asyncChannelB1</tp:ChannelId>
4111     </tp:ThisPartyActionBinding>
4112   </tp:CanSend>
4113   <tp:CanReceive>
4114     <tp:ThisPartyActionBinding
4115       tp:id="companyB_ABID4"
4116       tp:action="Purchase Order Request Action"
4117       tp:packageId="CompanyB_RequestPackage">
4118       <tp:BusinessTransactionCharacteristics
4119         tp:isNonRepudiationRequired="true"
4120         tp:isNonRepudiationReceiptRequired="true"
4121         tp:isConfidential="transient"
4122         tp:isAuthenticated="persistent"
4123         tp:isTamperProof="persistent"
4124         tp:isAuthorizationRequired="true"
4125         tp:timeToAcknowledgeReceipt="PT2H"
4126         tp:timeToPerform="P1D"/>
4127       <tp:ActionContext
4128         tp:binaryCollaboration="Request Purchase Order"
4129         tp:businessTransactionActivity="Request Purchase Order"
4130         tp:requestOrResponseAction="Purchase Order Request Action"/>

```

```

4131     <tp:ChannelId>asyncChannelB1</tp:ChannelId>
4132   </tp:ThisPartyActionBinding>
4133 </tp:CanReceive>
4134 <tp:CanReceive>
4135   <tp:ThisPartyActionBinding
4136     tp:id="companyB_ABID5" tp:action="ReceiptAcknowledgment"
4137     tp:packageId="CompanyB_ReceiptAcknowledgmentPackage">
4138     <tp:BusinessTransactionCharacteristics
4139       tp:isNonRepudiationRequired="true"
4140       tp:isNonRepudiationReceiptRequired="true"
4141       tp:isConfidential="transient"
4142       tp:isAuthenticated="persistent"
4143       tp:isTamperProof="persistent"
4144       tp:isAuthorizationRequired="true"/>
4145     <tp:ChannelId>asyncChannelB1</tp:ChannelId>
4146   </tp:ThisPartyActionBinding>
4147 </tp:CanReceive>
4148 <!-- The next binding uses a synchronous delivery channel -->
4149 <tp:CanReceive>
4150   <tp:ThisPartyActionBinding
4151     tp:id="companyB_ABID6"
4152     tp:action="Purchase Order Request Action"
4153     tp:packageId="CompanyB_SyncReplyPackage">
4154     <tp:BusinessTransactionCharacteristics
4155       tp:isNonRepudiationRequired="true"
4156       tp:isNonRepudiationReceiptRequired="true"
4157       tp:isConfidential="transient"
4158       tp:isAuthenticated="persistent"
4159       tp:isTamperProof="persistent"
4160       tp:isAuthorizationRequired="true"
4161       tp:timeToAcknowledgeReceipt="PT5M"
4162       tp:timeToPerform="PT5M"/>
4163     <tp:ActionContext
4164       tp:binaryCollaboration="Request Purchase Order"
4165       tp:businessTransactionActivity="Request Purchase Order"
4166       tp:requestOrResponseAction="Purchase Order Request Action"/>
4167     <tp:ChannelId>syncChannelB1</tp:ChannelId>
4168   </tp:ThisPartyActionBinding>
4169   <tp:CanSend>
4170     <tp:ThisPartyActionBinding
4171       tp:id="companyB_ABID7"
4172       tp:action="Purchase Order Confirmation Action"
4173       tp:packageId="CompanyB_ResponsePackage">
4174       <tp:BusinessTransactionCharacteristics
4175         tp:isNonRepudiationRequired="true"
4176         tp:isNonRepudiationReceiptRequired="true"
4177         tp:isConfidential="transient"
4178         tp:isAuthenticated="persistent"
4179         tp:isTamperProof="persistent"
4180         tp:isAuthorizationRequired="true"
4181         tp:timeToAcknowledgeReceipt="PT5M"/>
4182       <tp:ActionContext
4183         tp:binaryCollaboration="Request Purchase Order"
4184         tp:businessTransactionActivity="Request Purchase Order"
4185         tp:requestOrResponseAction="Purchase Order Confirmation Action"/>
4186       <tp:ChannelId>syncChannelB1</tp:ChannelId>
4187     </tp:ThisPartyActionBinding>
4188   </tp:CanSend>
4189 <tp:CanSend>
4190   <tp:ThisPartyActionBinding
4191     tp:id="companyB_ABID8"
4192     tp:action="Exception"
4193     tp:packageId="CompanyB_ExceptionPackage">
4194     <tp:BusinessTransactionCharacteristics
4195       tp:isNonRepudiationRequired="true"
4196       tp:isNonRepudiationReceiptRequired="true"
4197       tp:isConfidential="transient"
4198       tp:isAuthenticated="persistent"
4199       tp:isTamperProof="persistent"
4200       tp:isAuthorizationRequired="true"/>
4201     <tp:ChannelId>syncChannelB1</tp:ChannelId>
4202   </tp:ThisPartyActionBinding>
4203 </tp:CanSend>
4204 </tp:CanReceive>
4205 </tp:ServiceBinding>
4206 </tp:CollaborationRole>
4207 <!-- Certificates used by the "Seller" company -->
4208 <tp:Certificate tp:certId="CompanyB_AppCert">

```



```

4209     <ds:KeyInfo>
4210         <ds:KeyName>CompanyB_AppCert_Key</ds:KeyName>
4211     </ds:KeyInfo>
4212 </tp:Certificate>
4213 <tp:Certificate tp:certId="CompanyB_SigningCert">
4214     <ds:KeyInfo>
4215         <ds:KeyName>CompanyB_Signingcert_Key</ds:KeyName>
4216     </ds:KeyInfo>
4217 </tp:Certificate>
4218 <tp:Certificate tp:certId="CompanyB_EncryptionCert">
4219     <ds:KeyInfo>
4220         <ds:KeyName>CompanyB_EncryptionCert_Key</ds:KeyName>
4221     </ds:KeyInfo>
4222 </tp:Certificate>
4223 <tp:Certificate tp:certId="CompanyB_ServerCert">
4224     <ds:KeyInfo>
4225         <ds:KeyName>CompanyB_ServerCert_Key</ds:KeyName>
4226     </ds:KeyInfo>
4227 </tp:Certificate>
4228 <tp:Certificate tp:certId="CompanyB_ClientCert">
4229     <ds:KeyInfo>
4230         <ds:KeyName>CompanyB_ClientCert_Key</ds:KeyName>
4231     </ds:KeyInfo>
4232 </tp:Certificate>
4233 <tp:Certificate tp:certId="TrustedRootCertB4">
4234     <ds:KeyInfo>
4235         <ds:KeyName>TrustedRootCertB4_Key</ds:KeyName>
4236     </ds:KeyInfo>
4237 </tp:Certificate>
4238 <tp:Certificate tp:certId="TrustedRootCertB5">
4239     <ds:KeyInfo>
4240         <ds:KeyName>TrustedRootCertB5_Key</ds:KeyName>
4241     </ds:KeyInfo>
4242 </tp:Certificate>
4243 <tp:Certificate tp:certId="TrustedRootCertB6">
4244     <ds:KeyInfo>
4245         <ds:KeyName>TrustedRootCertB6_Key</ds:KeyName>
4246     </ds:KeyInfo>
4247 </tp:Certificate>
4248 <tp:Certificate tp:certId="TrustedRootCertB7">
4249     <ds:KeyInfo>
4250         <ds:KeyName>TrustedRootCertB7_Key</ds:KeyName>
4251     </ds:KeyInfo>
4252 </tp:Certificate>
4253 <tp:Certificate tp:certId="TrustedRootCertB8">
4254     <ds:KeyInfo>
4255         <ds:KeyName>TrustedRootCertB8_Key</ds:KeyName>
4256     </ds:KeyInfo>
4257 </tp:Certificate>
4258 <tp:SecurityDetails tp:securityId="CompanyB_TransportSecurity">
4259     <tp:TrustAnchors>
4260         <tp:AnchorCertificateRef tp:certId="TrustedRootCertB5"/>
4261         <tp:AnchorCertificateRef tp:certId="TrustedRootCertB6"/>
4262         <tp:AnchorCertificateRef tp:certId="TrustedRootCertB4"/>
4263     </tp:TrustAnchors>
4264 </tp:SecurityDetails>
4265 <tp:SecurityDetails tp:securityId="CompanyB_MessageSecurity">
4266     <tp:TrustAnchors>
4267         <tp:AnchorCertificateRef tp:certId="TrustedRootCertB8"/>
4268         <tp:AnchorCertificateRef tp:certId="TrustedRootCertB7"/>
4269     </tp:TrustAnchors>
4270 </tp:SecurityDetails>
4271 <!-- An asynchronous delivery channel -->
4272 <tp:DeliveryChannel
4273     tp:channelId="asyncChannelB1"
4274     tp:transportId="transportB1"
4275     tp:docExchangeId="docExchangeB1">
4276     <tp:MessagingCharacteristics
4277         tp:syncReplyMode="none"
4278         tp:ackRequested="always"
4279         tp:ackSignatureRequested="always"
4280         tp:duplicateElimination="always"/>
4281 </tp:DeliveryChannel>
4282 <!-- A synchronous delivery channel -->
4283 <tp:DeliveryChannel
4284     tp:channelId="syncChannelB1"
4285     tp:transportId="transportB2"
4286     tp:docExchangeId="docExchangeB1">

```

```

4287     <tp:MessagingCharacteristics
4288       tp:syncReplyMode="signalsAndResponse"
4289       tp:ackRequested="always"
4290       tp:ackSignatureRequested="always"
4291       tp:duplicateElimination="always"/>
4292   </tp:DeliveryChannel>
4293 <tp:Transport tp:transportId="transportB1">
4294   <tp:TransportSender>
4295     <tp:TransportProtocol tp:version="1.1">HTTP</tp:TransportProtocol>
4296     <tp:AccessAuthentication>basic</tp:AccessAuthentication>
4297     <tp:TransportClientSecurity>
4298       <tp:TransportSecurityProtocol tp:version="3.0">SSL</tp:TransportSecurityProtocol>
4299       <tp:ClientCertificateRef tp:certId="CompanyB_ClientCert"/>
4300       <tp:ServerSecurityDetailsRef tp:securityId="CompanyB_TransportSecurity"/>
4301     </tp:TransportClientSecurity>
4302   </tp:TransportSender>
4303   <tp:TransportReceiver>
4304     <tp:TransportProtocol tp:version="1.1">HTTP</tp:TransportProtocol>
4305     <tp:AccessAuthentication>basic</tp:AccessAuthentication>
4306     <tp:Endpoint
4307       tp:uri="https://www.CompanyB.com/servlets/ebxmlhandler/sync"
4308       tp:type="allPurpose"/>
4309     <tp:TransportServerSecurity>
4310       <tp:TransportSecurityProtocol tp:version="3.0">SSL</tp:TransportSecurityProtocol>
4311       <tp:ServerCertificateRef tp:certId="CompanyB_ServerCert"/>
4312       <tp:ClientSecurityDetailsRef tp:securityId="CompanyB_TransportSecurity"/>
4313     </tp:TransportServerSecurity>
4314   </tp:TransportReceiver>
4315 </tp:Transport>
4316 <tp:Transport tp:transportId="transportB2">
4317   <tp:TransportSender>
4318     <tp:TransportProtocol tp:version="1.1">HTTP</tp:TransportProtocol>
4319     <tp:AccessAuthentication>basic</tp:AccessAuthentication>
4320     <tp:TransportClientSecurity>
4321       <tp:TransportSecurityProtocol tp:version="3.0">SSL</tp:TransportSecurityProtocol>
4322       <tp:ClientCertificateRef tp:certId="CompanyB_ClientCert"/>
4323       <tp:ServerSecurityDetailsRef tp:securityId="CompanyB_TransportSecurity"/>
4324     </tp:TransportClientSecurity>
4325   </tp:TransportSender>
4326   <tp:TransportReceiver>
4327     <tp:TransportProtocol tp:version="1.1">HTTP</tp:TransportProtocol>
4328     <tp:AccessAuthentication>basic</tp:AccessAuthentication>
4329     <tp:Endpoint
4330       tp:uri="https://www.CompanyB.com/servlets/ebxmlhandler/async"
4331       tp:type="allPurpose"/>
4332     <tp:TransportServerSecurity>
4333       <tp:TransportSecurityProtocol tp:version="3.0">SSL</tp:TransportSecurityProtocol>
4334       <tp:ServerCertificateRef tp:certId="CompanyB_ServerCert"/>
4335       <tp:ClientSecurityDetailsRef tp:securityId="CompanyB_TransportSecurity"/>
4336     </tp:TransportServerSecurity>
4337   </tp:TransportReceiver>
4338 </tp:Transport>
4339 <tp:DocExchange tp:docExchangeId="docExchangeB1">
4340   <tp:ebXMLSenderBinding tp:version="2.0">
4341     <tp:ReliableMessaging>
4342       <tp:Retries>3</tp:Retries>
4343       <tp:RetryInterval>PT2H</tp:RetryInterval>
4344       <tp:MessageOrderSemantics>Guaranteed</tp:MessageOrderSemantics>
4345     </tp:ReliableMessaging>
4346     <tp:PersistDuration>P1D</tp:PersistDuration>
4347     <tp:SenderNonRepudiation>
4348       <tp:NonRepudiationProtocol>http://www.w3.org/2000/09/xmldsig#</tp:NonRepudiationProtocol>
4349       <tp:HashFunction>http://www.w3.org/2000/09/xmldsig#sha1</tp:HashFunction>
4350       <tp:SignatureAlgorithm>http://www.w3.org/2000/09/xmldsig#dsa-
4351       sha1</tp:SignatureAlgorithm>
4352       <tp:SigningCertificateRef tp:certId="CompanyB_SigningCert"/>
4353     </tp:SenderNonRepudiation>
4354     <tp:SenderDigitalEnvelope>
4355       <tp:DigitalEnvelopeProtocol tp:version="2.0">S/MIME</tp:DigitalEnvelopeProtocol>
4356       <tp:EncryptionAlgorithm>DES-CBC</tp:EncryptionAlgorithm>
4357       <tp:EncryptionSecurityDetailsRef tp:securityId="CompanyB_MessageSecurity"/>
4358     </tp:SenderDigitalEnvelope>
4359   </tp:ebXMLSenderBinding>
4360   <tp:ebXMLReceiverBinding tp:version="2.0">
4361     <tp:ReliableMessaging>
4362       <tp:Retries>3</tp:Retries>
4363       <tp:RetryInterval>PT2H</tp:RetryInterval>
4364

```

```

4365     <tp:MessageOrderSemantics>Guaranteed</tp:MessageOrderSemantics>
4366 </tp:ReliableMessaging>
4367     <tp:PersistDuration>PlD</tp:PersistDuration>
4368     <tp:ReceiverNonRepudiation>
4369
4370 <tp:NonRepudiationProtocol>http://www.w3.org/2000/09/xmldsig#</tp:NonRepudiationProtocol>
4371     <tp:HashFunction>http://www.w3.org/2000/09/xmldsig#sha1</tp:HashFunction>
4372     <tp:SignatureAlgorithm>http://www.w3.org/2000/09/xmldsig#dsa-
4373 sha1</tp:SignatureAlgorithm>
4374     <tp:SigningSecurityDetailsRef tp:securityId="CompanyB_MessageSecurity"/>
4375 </tp:ReceiverNonRepudiation>
4376     <tp:ReceiverDigitalEnvelope>
4377     <tp:DigitalEnvelopeProtocol tp:version="2.0">S/MIME</tp:DigitalEnvelopeProtocol>
4378     <tp:EncryptionAlgorithm>DES-CBC</tp:EncryptionAlgorithm>
4379     <tp:EncryptionCertificateRef tp:certId="CompanyB_EncryptionCert"/>
4380 </tp:ReceiverDigitalEnvelope>
4381 </tp:ebXMLReceiverBinding>
4382 </tp:DocExchange>
4383 </tp:PartyInfo>
4384 <!-- SimplePart corresponding to the SOAP Envelope -->
4385 <tp:SimplePart
4386   tp:id="CompanyB_MsgHdr"
4387   tp:mimetype="text/xml">
4388   <tp:NamespaceSupported
4389     tp:location="http://www.oasis-open.org/committees/ebxml-msg/schema/draft-msg-header-05.xsd"
4390     tp:version="2.0">
4391     http://www.oasis-open.org/committees/ebxml-msg/schema/draft-msg-header-05.xsd
4392   </tp:NamespaceSupported>
4393 </tp:SimplePart>
4394 <!-- SimplePart corresponding to a Receipt Acknowledgment business signal -->
4395 <tp:SimplePart
4396   tp:id="CompanyB_ReceiptAcknowledgment"
4397   tp:mimetype="application/xml">
4398   <tp:NamespaceSupported
4399     tp:location="http://www.ebxml.org/bpss/ReceiptAcknowledgment.xsd"
4400     tp:version="2.0">
4401     http://www.ebxml.org/bpss/ReceiptAcknowledgment.xsd
4402   </tp:NamespaceSupported>
4403 </tp:SimplePart>
4404 <!-- SimplePart corresponding to an Exception business signal -->
4405 <tp:SimplePart
4406   tp:id="CompanyB_Exception"
4407   tp:mimetype="application/xml">
4408   <tp:NamespaceSupported
4409     tp:location="http://www.oasis-open.org/committees/ebxml-msg/schema/draft-msg-header-05.xsd"
4410     tp:version="2.0">
4411     http://www.oasis-open.org/committees/ebxml-msg/schema/draft-msg-header-05.xsd
4412   </tp:NamespaceSupported>
4413 </tp:SimplePart>
4414 <!-- SimplePart corresponding to a request action -->
4415 <tp:SimplePart
4416   tp:id="CompanyB_Request"
4417   tp:mimetype="application/xml">
4418   <tp:NamespaceSupported
4419     tp:location="http://www.rosettanet.org/schemas/PIP3A4RequestPurchaseOrder.xsd"
4420     tp:version="2.0">
4421     http://www.rosettanet.org/schemas/PIP3A4RequestPurchaseOrder.xsd
4422   </tp:NamespaceSupported>
4423 </tp:SimplePart>
4424 <!-- SimplePart corresponding to a response action -->
4425 <tp:SimplePart
4426   tp:id="CompanyB_Response"
4427   tp:mimetype="application/xml">
4428   <tp:NamespaceSupported
4429     tp:location="http://www.rosettanet.org/schemas/PurchaseOrderConfirmation.xsd.xsd"
4430     tp:version="2.0">
4431     http://www.rosettanet.org/schemas/PIP3A4PurchaseOrderConfirmation.xsd
4432   </tp:NamespaceSupported>
4433 </tp:SimplePart>
4434 <!-- An ebXML message with a SOAP Envelope only -->
4435 <tp:Packaging tp:id="CompanyB_MshSignalPackage">
4436   <tp:ProcessingCapabilities
4437     tp:parse="true"
4438     tp:generate="true"/>
4439   <tp:CompositeList>
4440     <tp:Composite
4441       tp:id="CompanyB_MshSignal"
4442       tp:mimetype="multipart/related"

```

```

4443         tp:mimeparameters="type=text/xml">
4444         <tp:Constituent tp:idref="CompanyB_MsgHdr"/>
4445     </tp:Composite>
4446 </tp:CompositeList>
4447 </tp:Packaging>
4448 <!-- An ebXML message with a SOAP Envelope plus a request action payload -->
4449 <tp:Packaging tp:id="CompanyB_RequestPackage">
4450     <tp:ProcessingCapabilities
4451         tp:parse="true"
4452         tp:generate="true"/>
4453     <tp:CompositeList>
4454         <tp:Composite
4455             tp:id="RequestMsg"
4456             tp:mimetype="multipart/related"
4457             tp:mimeparameters="type=text/xml">
4458             <tp:Constituent tp:idref="CompanyB_MsgHdr"/>
4459             <tp:Constituent tp:idref="CompanyB_Request"/>
4460         </tp:Composite>
4461     </tp:CompositeList>
4462 </tp:Packaging>
4463 <!-- An ebXML message with a SOAP Envelope plus a response action payload -->
4464 <tp:Packaging tp:id="CompanyB_ResponsePackage">
4465     <tp:ProcessingCapabilities
4466         tp:parse="true"
4467         tp:generate="true"/>
4468     <tp:CompositeList>
4469         <tp:Composite
4470             tp:id="CompanyB_ResponseMsg"
4471             tp:mimetype="multipart/related"
4472             tp:mimeparameters="type=text/xml">
4473             <tp:Constituent tp:idref="CompanyB_MsgHdr"/>
4474             <tp:Constituent tp:idref="CompanyB_Response"/>
4475         </tp:Composite>
4476     </tp:CompositeList>
4477 </tp:Packaging>
4478 <!-- An ebXML message with a SOAP Envelope plus a Receipt Acknowledgment payload -->
4479 <tp:Packaging tp:id="CompanyB_ReceiptAcknowledgmentPackage">
4480     <tp:ProcessingCapabilities
4481         tp:parse="true"
4482         tp:generate="true"/>
4483     <tp:CompositeList>
4484         <tp:Composite
4485             tp:id="CompanyB_ReceiptAcknowledgmentMsg"
4486             tp:mimetype="multipart/related"
4487             tp:mimeparameters="type=text/xml">
4488             <tp:Constituent tp:idref="CompanyB_MsgHdr"/>
4489             <tp:Constituent tp:idref="CompanyB_ReceiptAcknowledgment"/>
4490         </tp:Composite>
4491     </tp:CompositeList>
4492 </tp:Packaging>
4493 <!-- An ebXML message with a SOAP Envelope plus an Exception payload -->
4494 <tp:Packaging tp:id="CompanyB_ExceptionPackage">
4495     <tp:ProcessingCapabilities
4496         tp:parse="true"
4497         tp:generate="true"/>
4498     <tp:CompositeList>
4499         <tp:Composite
4500             tp:id="CompanyB_ExceptionMsg"
4501             tp:mimetype="multipart/related"
4502             tp:mimeparameters="type=text/xml">
4503             <tp:Constituent tp:idref="CompanyB_MsgHdr"/>
4504             <tp:Constituent tp:idref="CompanyB_Exception"/>
4505         </tp:Composite>
4506     </tp:CompositeList>
4507 </tp:Packaging>
4508 <!-- An ebXML message with a Receipt Acknowledgment signal, plus a business response,
4509     or an ebXML message with an Exception signal -->
4510 <tp:Packaging tp:id="CompanyB_SyncReplyPackage">
4511     <tp:ProcessingCapabilities
4512         tp:parse="true"
4513         tp:generate="true"/>
4514     <tp:CompositeList>
4515         <tp:Composite
4516             tp:id="CompanyB_SignalAndResponseMsg"
4517             tp:mimetype="multipart/related"
4518             tp:mimeparameters="type=text/xml">
4519             <tp:Constituent tp:idref="CompanyB_MsgHdr"/>
4520             <tp:Constituent tp:idref="CompanyB_ReceiptAcknowledgment"/>

```

```
4521     <tp:Constituent tp:idref="CompanyB_Response"/>
4522     </tp:Composite>
4523 </tp:CompositeList>
4524 <tp:CompositeList>
4525     <tp:Composite
4526         tp:id="CompanyB_SyncExceptionMsg"
4527         tp:mimetype="multipart/related"
4528         tp:mimeparameters="type=text/xml">
4529         <tp:Constituent tp:idref="CompanyB_MsgHdr"/>
4530         <tp:Constituent tp:idref="CompanyB_Exception"/>
4531     </tp:Composite>
4532 </tp:CompositeList>
4533 </tp:Packaging>
4534 <tp:Comment xml:lang="en-US">Seller's Collaboration Protocol Profile</tp:Comment>
4535 </tp:CollaborationProtocolProfile>
4536
```

4537 **Appendix B Example of CPA Document (Non-Normative)**

4538 The example in this appendix is to be parsed with an XML Schema parser. The schema is
4539 available as an ASCII file at

4540 http://www.oasis-open.org/committees/ebxml-cppa/schema/cpp-cpa-2_0b.xsd

4541

4542 The example that can be parsed with the XSD is available at

4543 http://www.oasis-open.org/committees/ebxml-cppa/schema/cpa-example-2_0b.xml

4544

```

4545 <?xml version="1.0"?>
4546 <!-- Copyright UN/CEFACT and OASIS, 2001. All Rights Reserved. -->
4547 <tp:CollaborationProtocolAgreement
4548   xmlns:tp="http://www.oasis-open.org/committees/ebxml-cppa/schema/cpp-cpa-2_0.xsd"
4549   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4550   xmlns:xlink="http://www.w3.org/1999/xlink"
4551   xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
4552   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
4553   xsi:schemaLocation="http://www.oasis-open.org/committees/ebxml-cppa/schema/cpp-cpa-2_0.xsd
4554     cpp-cpa-2_0.xsd"
4555   tp:cpaid="uri:companyA-and-companyB-cpa" tp:version="2_0b">
4556   <tp:Status tp:value="proposed"/>
4557   <tp:Start>2001-05-20T07:21:00Z</tp:Start>
4558   <tp:End>2002-05-20T07:21:00Z</tp:End>
4559   <tp:ConversationConstraints tp:invocationLimit="100" tp:concurrentConversations="10"/>
4560   <!-- Party info for CompanyA -->
4561   <tp:PartyInfo
4562     tp:partyName="CompanyA"
4563     tp:defaultMshChannelId="asyncChannelA1"
4564     tp:defaultMshPackageId="CompanyA_MshSignalPackage">
4565     <tp:PartyId tp:type="urn:oasis:names:tc:ebxml-cppa:partyid-type:duns">123456789</tp:PartyId>
4566     <tp:PartyRef xlink:href="http://CompanyA.com/about.html"/>
4567     <tp:CollaborationRole>
4568       <tp:ProcessSpecification
4569         tp:version="2.0"
4570         tp:name="PIP3A4RequestPurchaseOrder"
4571         xlink:type="simple"
4572         xlink:href="http://www.rosettanet.org/processes/3A4.xml"
4573         tp:uuid="urn:icann:rosettanet.org:bpid:3A4$2.0"/>
4574       <tp:Role
4575         tp:name="Buyer"
4576         xlink:type="simple"
4577         xlink:href="http://www.rosettanet.org/processes/3A4.xml#Buyer"/>
4578       <tp:ApplicationCertificateRef tp:certId="CompanyA_AppCert"/>
4579       <tp:ServiceBinding>
4580         <tp:Service>bpid:icann:rosettanet.org:3A4$2.0</tp:Service>
4581         <tp:CanSend>
4582           <tp:ThisPartyActionBinding
4583             tp:id="companyA_ABID1"
4584             tp:action="Purchase Order Request Action"
4585             tp:packageId="CompanyA_RequestPackage">
4586             <tp:BusinessTransactionCharacteristics
4587               tp:isNonRepudiationRequired="true"
4588               tp:isNonRepudiationReceiptRequired="true"
4589               tp:isConfidential="transient"
4590               tp:isAuthenticated="persistent"
4591               tp:isTamperProof="persistent"
4592               tp:isAuthorizationRequired="true"
4593               tp:timeToAcknowledgeReceipt="PT2H"
4594               tp:timeToPerform="P1D"/>
4595             <tp>ActionContext
4596               tp:binaryCollaboration="Request Purchase Order"
4597               tp:businessTransactionActivity="Request Purchase Order"
4598               tp:requestOrResponseAction="Purchase Order Request Action"/>
4599             <tp:ChannelId>asyncChannelA1</tp:ChannelId>
4600           </tp:ThisPartyActionBinding>
4601           <tp:OtherPartyActionBinding>companyB_ABID4</tp:OtherPartyActionBinding>
4602         </tp:CanSend>
4603       <tp:CanSend>
4604         <tp:ThisPartyActionBinding
4605           tp:id="companyA_ABID2"

```

```

4606         tp:action="ReceiptAcknowledgement"
4607         tp:packageId="CompanyA_ReceiptAcknowledgmentPackage">
4608         <tp:BusinessTransactionCharacteristics
4609             tp:isNonRepudiationRequired="true"
4610             tp:isNonRepudiationReceiptRequired="true"
4611             tp:isConfidential="transient"
4612             tp:isAuthenticated="persistent"
4613             tp:isTamperProof="persistent"
4614             tp:isAuthorizationRequired="true"/>
4615         <tp:ChannelId>asyncChannelA1</tp:ChannelId>
4616     </tp:ThisPartyActionBinding>
4617     <tp:OtherPartyActionBinding>companyB_ABID5</tp:OtherPartyActionBinding>
4618 </tp:CanSend>
4619 <!-- The next binding uses a synchronous delivery channel -->
4620 <tp:CanSend>
4621     <tp:ThisPartyActionBinding
4622         tp:id="companyA_ABID6"
4623         tp:action="Purchase Order Request Action"
4624         tp:packageId="CompanyA_RequestPackage">
4625         <tp:BusinessTransactionCharacteristics
4626             tp:isNonRepudiationRequired="true"
4627             tp:isNonRepudiationReceiptRequired="true"
4628             tp:isConfidential="transient"
4629             tp:isAuthenticated="persistent"
4630             tp:isTamperProof="persistent"
4631             tp:isAuthorizationRequired="true"
4632             tp:timeToAcknowledgeReceipt="PT5M"
4633             tp:timeToPerform="PT5M"/>
4634         <tp>ActionContext
4635             tp:binaryCollaboration="Request Purchase Order"
4636             tp:businessTransactionActivity="Request Purchase Order"
4637             tp:requestOrResponseAction="Purchase Order Request Action"/>
4638         <tp:ChannelId>syncChannelA1</tp:ChannelId>
4639     </tp:ThisPartyActionBinding>
4640     <tp:OtherPartyActionBinding>companyB_ABID6</tp:OtherPartyActionBinding>
4641 <tp:CanReceive>
4642     <tp:ThisPartyActionBinding
4643         tp:id="companyA_ABID7"
4644         tp:action="Purchase Order Confirmation Action"
4645         tp:packageId="CompanyA_SyncReplyPackage">
4646         <tp:BusinessTransactionCharacteristics
4647             tp:isNonRepudiationRequired="true"
4648             tp:isNonRepudiationReceiptRequired="true"
4649             tp:isConfidential="transient"
4650             tp:isAuthenticated="persistent"
4651             tp:isTamperProof="persistent"
4652             tp:isAuthorizationRequired="true"
4653             tp:timeToAcknowledgeReceipt="PT5M"/>
4654         <tp>ActionContext
4655             tp:binaryCollaboration="Request Purchase Order"
4656             tp:businessTransactionActivity="Request Purchase Order"
4657             tp:requestOrResponseAction="Purchase Order Confirmation Action"/>
4658         <tp:ChannelId>syncChannelA1</tp:ChannelId>
4659     </tp:ThisPartyActionBinding>
4660     <tp:OtherPartyActionBinding>companyB_ABID7</tp:OtherPartyActionBinding>
4661 </tp:CanReceive>
4662 <tp:CanReceive>
4663     <tp:ThisPartyActionBinding
4664         tp:id="companyA_ABID8"
4665         tp:action="Exception"
4666         tp:packageId="CompanyA_ExceptionPackage">
4667         <tp:BusinessTransactionCharacteristics
4668             tp:isNonRepudiationRequired="true"
4669             tp:isNonRepudiationReceiptRequired="true"
4670             tp:isConfidential="transient"
4671             tp:isAuthenticated="persistent"
4672             tp:isTamperProof="persistent"
4673             tp:isAuthorizationRequired="true"/>
4674         <tp:ChannelId>syncChannelA1</tp:ChannelId>
4675     </tp:ThisPartyActionBinding>
4676     <tp:OtherPartyActionBinding>companyB_ABID8</tp:OtherPartyActionBinding>
4677 </tp:CanReceive>
4678 </tp:CanSend>
4679 <tp:CanReceive>
4680     <tp:ThisPartyActionBinding
4681         tp:id="companyA_ABID3"
4682         tp:action="Purchase Order Confirmation Action"
4683         tp:packageId="CompanyA_ResponsePackage">

```

```

4684     <tp:BusinessTransactionCharacteristics
4685         tp:isNonRepudiationRequired="true"
4686         tp:isNonRepudiationReceiptRequired="true"
4687         tp:isConfidential="transient"
4688         tp:isAuthenticated="persistent"
4689         tp:isTamperProof="persistent"
4690         tp:isAuthorizationRequired="true"
4691         tp:timeToAcknowledgeReceipt="PT2H" />
4692     <tp:ActionContext
4693         tp:binaryCollaboration="Request Purchase Order"
4694         tp:businessTransactionActivity="Request Purchase Order"
4695         tp:requestOrResponseAction="Purchase Order Confirmation Action" />
4696     <tp:ChannelId>asyncChannelA1</tp:ChannelId>
4697 </tp:ThisPartyActionBinding>
4698 <tp:OtherPartyActionBinding>companyB_ABID1</tp:OtherPartyActionBinding>
4699 </tp:CanReceive>
4700 <tp:CanReceive>
4701     <tp:ThisPartyActionBinding
4702         tp:id="companyA_ABID4"
4703         tp:action="ReceiptAcknowledgment"
4704         tp:packageId="CompanyA_ReceiptAcknowledgmentPackage">
4705         <tp:BusinessTransactionCharacteristics
4706             tp:isNonRepudiationRequired="true"
4707             tp:isNonRepudiationReceiptRequired="true"
4708             tp:isConfidential="transient"
4709             tp:isAuthenticated="persistent"
4710             tp:isTamperProof="persistent"
4711             tp:isAuthorizationRequired="true" />
4712         <tp:ChannelId>asyncChannelA1</tp:ChannelId>
4713     </tp:ThisPartyActionBinding>
4714     <tp:OtherPartyActionBinding>companyB_ABID2</tp:OtherPartyActionBinding>
4715 </tp:CanReceive>
4716 <tp:CanReceive>
4717     <tp:ThisPartyActionBinding
4718         tp:id="companyA_ABID5"
4719         tp:action="Exception"
4720         tp:packageId="CompanyA_ExceptionPackage">
4721         <tp:BusinessTransactionCharacteristics
4722             tp:isNonRepudiationRequired="true"
4723             tp:isNonRepudiationReceiptRequired="true"
4724             tp:isConfidential="transient"
4725             tp:isAuthenticated="persistent"
4726             tp:isTamperProof="persistent"
4727             tp:isAuthorizationRequired="true" />
4728         <tp:ChannelId>asyncChannelA1</tp:ChannelId>
4729     </tp:ThisPartyActionBinding>
4730     <tp:OtherPartyActionBinding>companyB_ABID3</tp:OtherPartyActionBinding>
4731 </tp:CanReceive>
4732 </tp:ServiceBinding>
4733 </tp:CollaborationRole>
4734 <!-- Certificates used by the "Buyer" company -->
4735 <tp:Certificate tp:certId="CompanyA_AppCert">
4736     <ds:KeyInfo>
4737         <ds:KeyName>CompanyA_AppCert_Key</ds:KeyName>
4738     </ds:KeyInfo>
4739 </tp:Certificate>
4740 <tp:Certificate tp:certId="CompanyA_SigningCert">
4741     <ds:KeyInfo>
4742         <ds:KeyName>CompanyA_SigningCert_Key</ds:KeyName>
4743     </ds:KeyInfo>
4744 </tp:Certificate>
4745 <tp:Certificate tp:certId="CompanyA_EncryptionCert">
4746     <ds:KeyInfo>
4747         <ds:KeyName>CompanyA_EncryptionCert_Key</ds:KeyName>
4748     </ds:KeyInfo>
4749 </tp:Certificate>
4750 <tp:Certificate tp:certId="CompanyA_ServerCert">
4751     <ds:KeyInfo>
4752         <ds:KeyName>CompanyA_ServerCert_Key</ds:KeyName>
4753     </ds:KeyInfo>
4754 </tp:Certificate>
4755 <tp:Certificate tp:certId="CompanyA_ClientCert">
4756     <ds:KeyInfo>
4757         <ds:KeyName>CompanyA_ClientCert_Key</ds:KeyName>
4758     </ds:KeyInfo>
4759 </tp:Certificate>
4760 <tp:Certificate tp:certId="TrustedRootCertA1">
4761     <ds:KeyInfo>

```



```

4762     <ds:KeyName>TrustedRootCertA1_Key </ds:KeyName>
4763     </ds:KeyInfo>
4764 </tp:Certificate>
4765 <tp:Certificate tp:certId="TrustedRootCertA2">
4766     <ds:KeyInfo>
4767         <ds:KeyName>TrustedRootCertA2_Key</ds:KeyName>
4768     </ds:KeyInfo>
4769 </tp:Certificate>
4770 <tp:Certificate tp:certId="TrustedRootCertA3">
4771     <ds:KeyInfo>
4772         <ds:KeyName>TrustedRootCertA3_Key</ds:KeyName>
4773     </ds:KeyInfo>
4774 </tp:Certificate>
4775 <tp:Certificate tp:certId="TrustedRootCertA4">
4776     <ds:KeyInfo>
4777         <ds:KeyName>TrustedRootCertA4_Key</ds:KeyName>
4778     </ds:KeyInfo>
4779 </tp:Certificate>
4780 <tp:Certificate tp:certId="TrustedRootCertA5">
4781     <ds:KeyInfo>
4782         <ds:KeyName>TrustedRootCertA5_Key</ds:KeyName>
4783     </ds:KeyInfo>
4784 </tp:Certificate>
4785 <tp:SecurityDetails tp:securityId="CompanyA_TransportSecurity">
4786     <tp:TrustAnchors>
4787         <tp:AnchorCertificateRef tp:certId="TrustedRootCertA1"/>
4788         <tp:AnchorCertificateRef tp:certId="TrustedRootCertA2"/>
4789         <tp:AnchorCertificateRef tp:certId="TrustedRootCertA4"/>
4790     </tp:TrustAnchors>
4791 </tp:SecurityDetails>
4792 <tp:SecurityDetails tp:securityId="CompanyA_MessageSecurity">
4793     <tp:TrustAnchors>
4794         <tp:AnchorCertificateRef tp:certId="TrustedRootCertA3"/>
4795         <tp:AnchorCertificateRef tp:certId="TrustedRootCertA5"/>
4796     </tp:TrustAnchors>
4797 </tp:SecurityDetails>
4798 <tp:DeliveryChannel
4800     tp:channelId="asyncChannelA1"
4801     tp:transportId="transportA1"
4802     tp:docExchangeId="docExchangeA1">
4803     <tp:MessagingCharacteristics
4804         tp:syncReplyMode="none"
4805         tp:ackRequested="always"
4806         tp:ackSignatureRequested="always"
4807         tp:duplicateElimination="always"/>
4808 </tp:DeliveryChannel>
4809 <tp:DeliveryChannel
4810     tp:channelId="syncChannelA1"
4811     tp:transportId="transportA2"
4812     tp:docExchangeId="docExchangeA1">
4813     <tp:MessagingCharacteristics
4814         tp:syncReplyMode="signalsAndResponse"
4815         tp:ackRequested="always"
4816         tp:ackSignatureRequested="always"
4817         tp:duplicateElimination="always"/>
4818 </tp:DeliveryChannel>
4819 <tp:Transport tp:transportId="transportA1">
4820     <tp:TransportSender>
4821         <tp:TransportProtocol tp:version="1.1">HTTP</tp:TransportProtocol>
4822         <tp:AccessAuthentication>basic</tp:AccessAuthentication>
4823         <tp:TransportClientSecurity>
4824             <tp:TransportSecurityProtocol tp:version="3.0">SSL</tp:TransportSecurityProtocol>
4825             <tp:ClientCertificateRef tp:certId="CompanyA_ClientCert"/>
4826             <tp:ServerSecurityDetailsRef tp:securityId="CompanyA_TransportSecurity"/>
4827         </tp:TransportClientSecurity>
4828     </tp:TransportSender>
4829     <tp:TransportReceiver>
4830         <tp:TransportProtocol tp:version="1.1">HTTP</tp:TransportProtocol>
4831         <tp:AccessAuthentication>basic</tp:AccessAuthentication>
4832         <tp:Endpoint
4833             tp:uri="https://www.CompanyA.com/servlets/ebxmlhandler/async"
4834             tp:type="allPurpose"/>
4835         <tp:TransportServerSecurity>
4836             <tp:TransportSecurityProtocol tp:version="3.0">SSL</tp:TransportSecurityProtocol>
4837             <tp:ServerCertificateRef tp:certId="CompanyA_ServerCert"/>
4838             <tp:ClientSecurityDetailsRef tp:securityId="CompanyA_TransportSecurity"/>
4839         </tp:TransportServerSecurity>
4839     </tp:TransportReceiver>

```

```

4840 </tp:Transport>
4841 <tp:Transport tp:transportId="transportA2">
4842 <tp:TransportSender>
4843 <tp:TransportProtocol tp:version="1.1">HTTP</tp:TransportProtocol>
4844 <tp:AccessAuthentication>basic</tp:AccessAuthentication>
4845 <tp:TransportClientSecurity>
4846 <tp:TransportSecurityProtocol tp:version="3.0">SSL</tp:TransportSecurityProtocol>
4847 <tp:ClientCertificateRef tp:certId="CompanyA_ClientCert"/>
4848 <tp:ServerSecurityDetailsRef tp:securityId="CompanyA_TransportSecurity"/>
4849 </tp:TransportClientSecurity>
4850 </tp:TransportSender>
4851 <tp:TransportReceiver>
4852 <tp:TransportProtocol tp:version="1.1">HTTP</tp:TransportProtocol>
4853 <tp:AccessAuthentication>basic</tp:AccessAuthentication>
4854 <tp:Endpoint>
4855 tp:uri="https://www.CompanyA.com/servlets/ebxmlhandler/sync"
4856 tp:type="allPurpose"/>
4857 <tp:TransportServerSecurity>
4858 <tp:TransportSecurityProtocol tp:version="3.0">SSL</tp:TransportSecurityProtocol>
4859 <tp:ServerCertificateRef tp:certId="CompanyA_ServerCert"/>
4860 <tp:ClientSecurityDetailsRef tp:securityId="CompanyA_TransportSecurity"/>
4861 </tp:TransportServerSecurity>
4862 </tp:TransportReceiver>
4863 </tp:Transport>
4864 <tp:DocExchange tp:docExchangeId="docExchangeA1">
4865 <tp:ebXMLSenderBinding tp:version="2.0">
4866 <tp:ReliableMessaging>
4867 <tp:Retries>3</tp:Retries>
4868 <tp:RetryInterval>PT2H</tp:RetryInterval>
4869 <tp:MessageOrderSemantics>Guaranteed</tp:MessageOrderSemantics>
4870 </tp:ReliableMessaging>
4871 <tp:PersistDuration>PlD</tp:PersistDuration>
4872 <tp:SenderNonRepudiation>
4873 <tp:NonRepudiationProtocol>http://www.w3.org/2000/09/xmldsig#</tp:NonRepudiationProtocol>
4874 <tp:HashFunction>http://www.w3.org/2000/09/xmldsig#sha1</tp:HashFunction>
4875 <tp:SignatureAlgorithm>http://www.w3.org/2000/09/xmldsig#dsa-
4876 sha1</tp:SignatureAlgorithm>
4877 <tp:SigningCertificateRef tp:certId="CompanyA_SigningCert"/>
4878 </tp:SenderNonRepudiation>
4879 <tp:SenderDigitalEnvelope>
4880 <tp:DigitalEnvelopeProtocol tp:version="2.0">S/MIME</tp:DigitalEnvelopeProtocol>
4881 <tp:EncryptionAlgorithm>DES-CBC</tp:EncryptionAlgorithm>
4882 <tp:EncryptionSecurityDetailsRef tp:securityId="CompanyA_MessageSecurity"/>
4883 </tp:SenderDigitalEnvelope>
4884 </tp:ebXMLSenderBinding>
4885 <tp:ebXMLReceiverBinding tp:version="2.0">
4886 <tp:ReliableMessaging>
4887 <tp:Retries>3</tp:Retries>
4888 <tp:RetryInterval>PT2H</tp:RetryInterval>
4889 <tp:MessageOrderSemantics>Guaranteed</tp:MessageOrderSemantics>
4890 </tp:ReliableMessaging>
4891 <tp:PersistDuration>PlD</tp:PersistDuration>
4892 <tp:ReceiverNonRepudiation>
4893 <tp:NonRepudiationProtocol>http://www.w3.org/2000/09/xmldsig#</tp:NonRepudiationProtocol>
4894 <tp:HashFunction>http://www.w3.org/2000/09/xmldsig#sha1</tp:HashFunction>
4895 <tp:SignatureAlgorithm>http://www.w3.org/2000/09/xmldsig#dsa-
4896 sha1</tp:SignatureAlgorithm>
4897 <tp:SigningSecurityDetailsRef tp:securityId="CompanyA_MessageSecurity"/>
4898 </tp:ReceiverNonRepudiation>
4899 <tp:ReceiverDigitalEnvelope>
4900 <tp:DigitalEnvelopeProtocol tp:version="2.0">S/MIME</tp:DigitalEnvelopeProtocol>
4901 <tp:EncryptionAlgorithm>DES-CBC</tp:EncryptionAlgorithm>
4902 <tp:EncryptionCertificateRef tp:certId="CompanyA_EncryptionCert"/>
4903 </tp:ReceiverDigitalEnvelope>
4904 </tp:ebXMLReceiverBinding>
4905 </tp:DocExchange>
4906 </tp:PartyInfo>
4907 <!-- Party info for CompanyB -->
4908 <tp:PartyInfo
4909 tp:partyName="CompanyB"
4910 tp:defaultMshChannelId="asyncChannelB1"
4911 tp:defaultMshPackageId="CompanyB_MshSignalPackage">
4912 <tp:PartyId tp:type="urn:oasis:names:tc:ebxml-cppa:partyid-type:duns">987654321</tp:PartyId>
4913 <tp:PartyRef xlink:type="simple" xlink:href="http://CompanyB.com/about.html"/>
4914 <tp:CollaborationRole>
4915 <tp:ProcessSpecification

```

```

4918     tp:version="2.0"
4919     tp:name="PIP3A4RequestPurchaseOrder"
4920     xlink:type="simple"
4921     xlink:href="http://www.rosettanet.org/processes/3A4.xml"
4922     tp:uuid="urn:icann:rosettanet.org:bpid:3A4$2.0"/>
4923 <tp:Role
4924   tp:name="Seller"
4925   xlink:type="simple"
4926   xlink:href="http://www.rosettanet.org/processes/3A4.xml#seller"/>
4927 <tp:ApplicationCertificateRef tp:certId="CompanyB_AppCert"/>
4928 <tp:ServiceBinding>
4929   <tp:Service>bpid:icann:rosettanet.org:3A4$2.0</tp:Service>
4930   <tp:CanSend>
4931     <tp:ThisPartyActionBinding
4932       tp:id="companyB_ABID1"
4933       tp:action="Purchase Order Confirmation Action"
4934       tp:packageId="CompanyB_ResponsePackage">
4935       <tp:BusinessTransactionCharacteristics
4936         tp:isNonRepudiationRequired="true"
4937         tp:isNonRepudiationReceiptRequired="true"
4938         tp:isConfidential="transient"
4939         tp:isAuthenticated="persistent"
4940         tp:isTamperProof="persistent"
4941         tp:isAuthorizationRequired="true"
4942         tp:timeToAcknowledgeReceipt="PT2H"/>
4943       <tp>ActionContext
4944         tp:binaryCollaboration="Request Purchase Order"
4945         tp:businessTransactionActivity="Request Purchase Order"
4946         tp:requestOrResponseAction="Purchase Order Confirmation Action"/>
4947       <tp:ChannelId>asyncChannelB1</tp:ChannelId>
4948     </tp:ThisPartyActionBinding>
4949     <tp:OtherPartyActionBinding>companyA_ABID3</tp:OtherPartyActionBinding>
4950   </tp:CanSend>
4951   <tp:CanSend>
4952     <tp:ThisPartyActionBinding
4953       tp:id="companyB_ABID2"
4954       tp:action="ReceiptAcknowledgement"
4955       tp:packageId="CompanyB_ReceiptAcknowledgmentPackage">
4956       <tp:BusinessTransactionCharacteristics
4957         tp:isNonRepudiationRequired="true"
4958         tp:isNonRepudiationReceiptRequired="true"
4959         tp:isConfidential="transient"
4960         tp:isAuthenticated="persistent"
4961         tp:isTamperProof="persistent"
4962         tp:isAuthorizationRequired="true"/>
4963       <tp:ChannelId>asyncChannelB1</tp:ChannelId>
4964     </tp:ThisPartyActionBinding>
4965     <tp:OtherPartyActionBinding>companyA_ABID4</tp:OtherPartyActionBinding>
4966   </tp:CanSend>
4967   <tp:CanSend>
4968     <tp:ThisPartyActionBinding
4969       tp:id="companyB_ABID3"
4970       tp:action="Exception"
4971       tp:packageId="CompanyB_ExceptionPackage">
4972       <tp:BusinessTransactionCharacteristics
4973         tp:isNonRepudiationRequired="true"
4974         tp:isNonRepudiationReceiptRequired="true"
4975         tp:isConfidential="transient"
4976         tp:isAuthenticated="persistent"
4977         tp:isTamperProof="persistent"
4978         tp:isAuthorizationRequired="true"/>
4979       <tp:ChannelId>asyncChannelB1</tp:ChannelId>
4980     </tp:ThisPartyActionBinding>
4981     <tp:OtherPartyActionBinding>companyA_ABID5</tp:OtherPartyActionBinding>
4982   </tp:CanSend>
4983   <tp:CanReceive>
4984     <tp:ThisPartyActionBinding
4985       tp:id="companyB_ABID4"
4986       tp:action="Purchase Order Request Action"
4987       tp:packageId="CompanyB_RequestPackage">
4988       <tp:BusinessTransactionCharacteristics
4989         tp:isNonRepudiationRequired="true"
4990         tp:isNonRepudiationReceiptRequired="true"
4991         tp:isConfidential="transient"
4992         tp:isAuthenticated="persistent"
4993         tp:isTamperProof="persistent"
4994         tp:isAuthorizationRequired="true"

```

```

4996         tp:timeToAcknowledgeReceipt="PT2H"
4997         tp:timeToPerform="P1D" />
4998     <tp:ActionContext
4999         tp:binaryCollaboration="Request Purchase Order"
5000         tp:businessTransactionActivity="Request Purchase Order"
5001         tp:requestOrResponseAction="Purchase Order Request Action" />
5002     <tp:ChannelId>asyncChannelB1</tp:ChannelId>
5003 </tp:ThisPartyActionBinding>
5004 <tp:OtherPartyActionBinding>companyA_ABID1</tp:OtherPartyActionBinding>
5005 </tp:CanReceive>
5006 <tp:CanReceive>
5007     <tp:ThisPartyActionBinding
5008         tp:id="companyB_ABID5"
5009         tp:action="ReceiptAcknowledgment"
5010         tp:packageId="CompanyB_ReceiptAcknowledgmentPackage">
5011     <tp:BusinessTransactionCharacteristics
5012         tp:isNonRepudiationRequired="true"
5013         tp:isNonRepudiationReceiptRequired="true"
5014         tp:isConfidential="transient"
5015         tp:isAuthenticated="persistent"
5016         tp:isTamperProof="persistent"
5017         tp:isAuthorizationRequired="true" />
5018     <tp:ChannelId>asyncChannelB1</tp:ChannelId>
5019 </tp:ThisPartyActionBinding>
5020 <tp:OtherPartyActionBinding>companyA_ABID2</tp:OtherPartyActionBinding>
5021 </tp:CanReceive>
5022 <!-- The next binding uses a synchronous delivery channel -->
5023 <tp:CanReceive>
5024     <tp:ThisPartyActionBinding
5025         tp:id="companyB_ABID6"
5026         tp:action="Purchase Order Request Action"
5027         tp:packageId="CompanyB_RequestPackage">
5028     <tp:BusinessTransactionCharacteristics
5029         tp:isNonRepudiationRequired="true"
5030         tp:isNonRepudiationReceiptRequired="true"
5031         tp:isConfidential="transient"
5032         tp:isAuthenticated="persistent"
5033         tp:isTamperProof="persistent"
5034         tp:isAuthorizationRequired="true"
5035         tp:timeToAcknowledgeReceipt="PT5M"
5036         tp:timeToPerform="PT5M" />
5037     <tp:ActionContext
5038         tp:binaryCollaboration="Request Purchase Order"
5039         tp:businessTransactionActivity="Request Purchase Order"
5040         tp:requestOrResponseAction="Purchase Order Request Action" />
5041     <tp:ChannelId>syncChannelB1</tp:ChannelId>
5042 </tp:ThisPartyActionBinding>
5043 <tp:OtherPartyActionBinding>companyA_ABID6</tp:OtherPartyActionBinding>
5044 <tp:CanSend>
5045     <tp:ThisPartyActionBinding
5046         tp:id="companyB_ABID7"
5047         tp:action="Purchase Order Confirmation Action"
5048         tp:packageId="CompanyB_SyncReplyPackage">
5049     <tp:BusinessTransactionCharacteristics
5050         tp:isNonRepudiationRequired="true"
5051         tp:isNonRepudiationReceiptRequired="true"
5052         tp:isConfidential="transient"
5053         tp:isAuthenticated="persistent"
5054         tp:isTamperProof="persistent"
5055         tp:isAuthorizationRequired="true"
5056         tp:timeToAcknowledgeReceipt="PT5M" />
5057     <tp:ActionContext
5058         tp:binaryCollaboration="Request Purchase Order"
5059         tp:businessTransactionActivity="Request Purchase Order"
5060         tp:requestOrResponseAction="Purchase Order Confirmation Action" />
5061     <tp:ChannelId>syncChannelB1</tp:ChannelId>
5062 </tp:ThisPartyActionBinding>
5063 <tp:OtherPartyActionBinding>companyA_ABID7</tp:OtherPartyActionBinding>
5064 </tp:CanSend>
5065 <tp:CanSend>
5066     <tp:ThisPartyActionBinding
5067         tp:id="companyB_ABID8"
5068         tp:action="Exception"
5069         tp:packageId="CompanyB_ExceptionPackage">
5070     <tp:BusinessTransactionCharacteristics
5071         tp:isNonRepudiationRequired="true"
5072         tp:isNonRepudiationReceiptRequired="true"
5073         tp:isConfidential="transient"

```

```

5074         tp:isAuthenticated="persistent"
5075         tp:isTamperProof="persistent"
5076         tp:isAuthorizationRequired="true" />
5077     <tp:ChannelId>syncChannelB1</tp:ChannelId>
5078 </tp:ThisPartyActionBinding>
5079 <tp:OtherPartyActionBinding>companyA_ABID8</tp:OtherPartyActionBinding>
5080 </tp:CanSend>
5081 </tp:CanReceive>
5082 </tp:ServiceBinding>
5083 </tp:CollaborationRole>
5084 <!-- Certificates used by the "Seller" company -->
5085 <tp:Certificate tp:certId="CompanyB_AppCert">
5086     <ds:KeyInfo>
5087         <ds:KeyName>CompanyB_AppCert_Key</ds:KeyName>
5088     </ds:KeyInfo>
5089 </tp:Certificate>
5090 <tp:Certificate tp:certId="CompanyB_SigningCert">
5091     <ds:KeyInfo>
5092         <ds:KeyName>CompanyB_Signingcert_Key</ds:KeyName>
5093     </ds:KeyInfo>
5094 </tp:Certificate>
5095 <tp:Certificate tp:certId="CompanyB_EncryptionCert">
5096     <ds:KeyInfo>
5097         <ds:KeyName>CompanyB_EncryptionCert_Key</ds:KeyName>
5098     </ds:KeyInfo>
5099 </tp:Certificate>
5100 <tp:Certificate tp:certId="CompanyB_ServerCert">
5101     <ds:KeyInfo>
5102         <ds:KeyName>CompanyB_ServerCert_Key</ds:KeyName>
5103     </ds:KeyInfo>
5104 </tp:Certificate>
5105 <tp:Certificate tp:certId="CompanyB_ClientCert">
5106     <ds:KeyInfo>
5107         <ds:KeyName>CompanyB_ClientCert_Key</ds:KeyName>
5108     </ds:KeyInfo>
5109 </tp:Certificate>
5110 <tp:Certificate tp:certId="TrustedRootCertB4">
5111     <ds:KeyInfo>
5112         <ds:KeyName>TrustedRootCertB4_Key</ds:KeyName>
5113     </ds:KeyInfo>
5114 </tp:Certificate>
5115 <tp:Certificate tp:certId="TrustedRootCertB5">
5116     <ds:KeyInfo>
5117         <ds:KeyName>TrustedRootCertB5_Key</ds:KeyName>
5118     </ds:KeyInfo>
5119 </tp:Certificate>
5120 <tp:Certificate tp:certId="TrustedRootCertB6">
5121     <ds:KeyInfo>
5122         <ds:KeyName>TrustedRootCertB6_Key</ds:KeyName>
5123     </ds:KeyInfo>
5124 </tp:Certificate>
5125 <tp:Certificate tp:certId="TrustedRootCertB7">
5126     <ds:KeyInfo>
5127         <ds:KeyName>TrustedRootCertB7_Key</ds:KeyName>
5128     </ds:KeyInfo>
5129 </tp:Certificate>
5130 <tp:Certificate tp:certId="TrustedRootCertB8">
5131     <ds:KeyInfo>
5132         <ds:KeyName>TrustedRootCertB8_Key</ds:KeyName>
5133     </ds:KeyInfo>
5134 </tp:Certificate>
5135 <tp:SecurityDetails tp:securityId="CompanyB_TransportSecurity">
5136     <tp:TrustAnchors>
5137         <tp:AnchorCertificateRef tp:certId="TrustedRootCertB5" />
5138         <tp:AnchorCertificateRef tp:certId="TrustedRootCertB6" />
5139         <tp:AnchorCertificateRef tp:certId="TrustedRootCertB4" />
5140     </tp:TrustAnchors>
5141 </tp:SecurityDetails>
5142 <tp:SecurityDetails tp:securityId="CompanyB_MessageSecurity">
5143     <tp:TrustAnchors>
5144         <tp:AnchorCertificateRef tp:certId="TrustedRootCertB8" />
5145         <tp:AnchorCertificateRef tp:certId="TrustedRootCertB7" />
5146     </tp:TrustAnchors>
5147 </tp:SecurityDetails>
5148 <!-- An asynchronous delivery channel -->
5149 <tp:DeliveryChannel
5150     tp:channelId="asyncChannelB1"
5151     tp:transportId="transportB1"

```

```

51 tp:docExchangeId="docExchangeB1">
52 <tp:MessagingCharacteristics
53   tp:syncReplyMode="none"
54   tp:ackRequested="always"
55   tp:ackSignatureRequested="always"
56   tp:duplicateElimination="always"/>
57 </tp:DeliveryChannel>
58 <!-- A synchronous delivery channel -->
59 <tp:DeliveryChannel
60   tp:channelId="syncChannelB1"
61   tp:transportId="transportB2"
62   tp:docExchangeId="docExchangeB1">
63   <tp:MessagingCharacteristics
64     tp:syncReplyMode="signalsAndResponse"
65     tp:ackRequested="always"
66     tp:ackSignatureRequested="always"
67     tp:duplicateElimination="always"/>
68 </tp:DeliveryChannel>
69 <tp:Transport tp:transportId="transportB1">
70   <tp:TransportSender>
71     <tp:TransportProtocol tp:version="1.1">HTTP</tp:TransportProtocol>
72     <tp:AccessAuthentication>basic</tp:AccessAuthentication>
73     <tp:TransportClientSecurity>
74       <tp:TransportSecurityProtocol tp:version="3.0">SSL</tp:TransportSecurityProtocol>
75       <tp:ClientCertificateRef tp:certId="CompanyB_ClientCert"/>
76       <tp:ServerSecurityDetailsRef tp:securityId="CompanyB_TransportSecurity"/>
77     </tp:TransportClientSecurity>
78   </tp:TransportSender>
79   <tp:TransportReceiver>
80     <tp:TransportProtocol tp:version="1.1">HTTP</tp:TransportProtocol>
81     <tp:AccessAuthentication>basic</tp:AccessAuthentication>
82     <tp:Endpoint
83       tp:uri="https://www.CompanyB.com/servlets/ebxmlhandler/async"
84       tp:type="allPurpose"/>
85     <tp:TransportServerSecurity>
86       <tp:TransportSecurityProtocol tp:version="3.0">SSL</tp:TransportSecurityProtocol>
87       <tp:ServerCertificateRef tp:certId="CompanyB_ServerCert"/>
88       <tp:ClientSecurityDetailsRef tp:securityId="CompanyB_TransportSecurity"/>
89     </tp:TransportServerSecurity>
90   </tp:TransportReceiver>
91 </tp:Transport>
92 <tp:Transport tp:transportId="transportB2">
93   <tp:TransportSender>
94     <tp:TransportProtocol tp:version="1.1">HTTP</tp:TransportProtocol>
95     <tp:AccessAuthentication>basic</tp:AccessAuthentication>
96     <tp:TransportClientSecurity>
97       <tp:TransportSecurityProtocol tp:version="3.0">SSL</tp:TransportSecurityProtocol>
98       <tp:ClientCertificateRef tp:certId="CompanyB_ClientCert"/>
99       <tp:ServerSecurityDetailsRef tp:securityId="CompanyB_TransportSecurity"/>
100     </tp:TransportClientSecurity>
101   </tp:TransportSender>
102   <tp:TransportReceiver>
103     <tp:TransportProtocol tp:version="1.1">HTTP</tp:TransportProtocol>
104     <tp:AccessAuthentication>basic</tp:AccessAuthentication>
105     <tp:Endpoint
106       tp:uri="https://www.CompanyB.com/servlets/ebxmlhandler/sync"
107       tp:type="allPurpose"/>
108     <tp:TransportServerSecurity>
109       <tp:TransportSecurityProtocol tp:version="3.0">SSL</tp:TransportSecurityProtocol>
110       <tp:ServerCertificateRef tp:certId="CompanyB_ServerCert"/>
111       <tp:ClientSecurityDetailsRef tp:securityId="CompanyB_TransportSecurity"/>
112     </tp:TransportServerSecurity>
113   </tp:TransportReceiver>
114 </tp:Transport>
115 <tp:DocExchange tp:docExchangeId="docExchangeB1">
116   <tp:ebXMLSenderBinding tp:version="2.0">
117     <tp:ReliableMessaging>
118       <tp:Retries>3</tp:Retries>
119       <tp:RetryInterval>PT2H</tp:RetryInterval>
120       <tp:MessageOrderSemantics>Guaranteed</tp:MessageOrderSemantics>
121     </tp:ReliableMessaging>
122     <tp:PersistDuration>P1D</tp:PersistDuration>
123     <tp:SenderNonRepudiation>
124       <tp:NonRepudiationProtocol>http://www.w3.org/2000/09/xmldsig#</tp:NonRepudiationProtocol>
125       <tp:HashFunction>http://www.w3.org/2000/09/xmldsig#sha1</tp:HashFunction>
126       <tp:SignatureAlgorithm>http://www.w3.org/2000/09/xmldsig#dsa-
127       sha1</tp:SignatureAlgorithm>

```

```

5230      <tp:SigningCertificateRef tp:certId="CompanyB_SigningCert" />
5231    </tp:SenderNonRepudiation>
5232    <tp:SenderDigitalEnvelope>
5233      <tp:DigitalEnvelopeProtocol tp:version="2.0">S/MIME</tp:DigitalEnvelopeProtocol>
5234      <tp:EncryptionAlgorithm>DES-CBC</tp:EncryptionAlgorithm>
5235      <tp:EncryptionSecurityDetailsRef tp:securityId="CompanyB_MessageSecurity" />
5236    </tp:SenderDigitalEnvelope>
5237  </tp:ebXMLSenderBinding>
5238  <tp:ebXMLReceiverBinding tp:version="2.0">
5239    <tp:ReliableMessaging>
5240      <tp:Retries>3</tp:Retries>
5241      <tp:RetryInterval>PT2H</tp:RetryInterval>
5242      <tp:MessageOrderSemantics>Guaranteed</tp:MessageOrderSemantics>
5243    </tp:ReliableMessaging>
5244    <tp:PersistDuration>P1D</tp:PersistDuration>
5245    <tp:ReceiverNonRepudiation>
5246  </tp:NonRepudiationProtocol>http://www.w3.org/2000/09/xmldsig#</tp:NonRepudiationProtocol>
5247    <tp:HashFunction>http://www.w3.org/2000/09/xmldsig#sha1</tp:HashFunction>
5248    <tp:SignatureAlgorithm>http://www.w3.org/2000/09/xmldsig#dsa-
5249  sha1</tp:SignatureAlgorithm>
5250    <tp:SigningSecurityDetailsRef tp:securityId="CompanyB_MessageSecurity" />
5251  </tp:ReceiverNonRepudiation>
5252  <tp:ReceiverDigitalEnvelope>
5253    <tp:DigitalEnvelopeProtocol tp:version="2.0">S/MIME</tp:DigitalEnvelopeProtocol>
5254    <tp:EncryptionAlgorithm>DES-CBC</tp:EncryptionAlgorithm>
5255    <tp:EncryptionCertificateRef tp:certId="CompanyB_EncryptionCert" />
5256  </tp:ReceiverDigitalEnvelope>
5257  </tp:ebXMLReceiverBinding>
5258  </tp:DocExchange>
5259 </tp:PartyInfo>
5260 <!-- SimplePart corresponding to the SOAP Envelope -->
5261 <tp:SimplePart
5262   tp:id="CompanyA_MsgHdr"
5263   tp:mimetype="text/xml">
5264   <tp:NamespaceSupported
5265     tp:location="http://www.oasis-open.org/committees/ebxml-msg/schema/msg-header-2_0.xsd"
5266     tp:version="2.0">
5267     http://www.oasis-open.org/committees/ebxml-msg/schema/msg-header-2_0.xsd
5268   </tp:NamespaceSupported>
5269 </tp:SimplePart>
5270 <tp:SimplePart
5271   tp:id="CompanyB_MsgHdr"
5272   tp:mimetype="text/xml">
5273   <tp:NamespaceSupported
5274     tp:location="http://www.oasis-open.org/committees/ebxml-msg/schema/msg-header-2_0.xsd"
5275     tp:version="2.0">
5276     http://www.oasis-open.org/committees/ebxml-msg/schema/msg-header-2_0.xsd
5277   </tp:NamespaceSupported>
5278 </tp:SimplePart>
5279 <!-- SimplePart corresponding to a Receipt Acknowledgment business signal -->
5280 <tp:SimplePart
5281   tp:id="CompanyA_ReceiptAcknowledgment"
5282   tp:mimetype="application/xml">
5283   <tp:NamespaceSupported
5284     tp:location="http://www.ebxml.org/bpss/ReceiptAcknowledgment.xsd"
5285     tp:version="2.0">http://www.ebxml.org/bpss/ReceiptAcknowledgment.xsd
5286   </tp:NamespaceSupported>
5287 </tp:SimplePart>
5288 <tp:SimplePart
5289   tp:id="CompanyB_ReceiptAcknowledgment"
5290   tp:mimetype="application/xml">
5291   <tp:NamespaceSupported
5292     tp:location="http://www.ebxml.org/bpss/ReceiptAcknowledgment.xsd"
5293     tp:version="2.0">
5294     http://www.ebxml.org/bpss/ReceiptAcknowledgment.xsd
5295   </tp:NamespaceSupported>
5296 </tp:SimplePart>
5297 <!-- SimplePart corresponding to an Exception business signal -->
5298 <tp:SimplePart
5299   tp:id="CompanyA_Exception"
5300   tp:mimetype="application/xml">
5301   <tp:NamespaceSupported
5302     tp:location="http://www.oasis-open.org/committees/ebxml-msg/schema/msg-header-2_0.xsd"
5303     tp:version="2.0">
5304     http://www.oasis-open.org/committees/ebxml-msg/schema/msg-header-2_0.xsd
5305   </tp:NamespaceSupported>
5306 </tp:SimplePart>
5307

```

```

5308 <tp:SimplePart
5309   tp:id="CompanyB_Exception"
5310   tp:mimetype="application/xml">
5311   <tp:NamespaceSupported
5312     tp:location="http://www.oasis-open.org/committees/ebxml-msg/schema/msg-header-2_0.xsd"
5313     tp:version="2.0">
5314     http://www.oasis-open.org/committees/ebxml-msg/schema/msg-header-2_0.xsd
5315   </tp:NamespaceSupported>
5316 </tp:SimplePart>
5317 <!-- SimplePart corresponding to a request action -->
5318 <tp:SimplePart
5319   tp:id="CompanyA_Request"
5320   tp:mimetype="application/xml">
5321   <tp:NamespaceSupported
5322     tp:location="http://www.rosettanet.org/schemas/PIP3A4RequestPurchaseOrder.xsd"
5323     tp:version="1.0">
5324     http://www.rosettanet.org/schemas/PIP3A4RequestPurchaseOrder.xsd
5325   </tp:NamespaceSupported>
5326 </tp:SimplePart>
5327 <tp:SimplePart
5328   tp:id="CompanyB_Request"
5329   tp:mimetype="application/xml">
5330   <tp:NamespaceSupported
5331     tp:location="http://www.rosettanet.org/schemas/PIP3A4RequestPurchaseOrder.xsd"
5332     tp:version="1.0">
5333     http://www.rosettanet.org/schemas/PIP3A4RequestPurchaseOrder.xsd
5334   </tp:NamespaceSupported>
5335 </tp:SimplePart>
5336 <!-- SimplePart corresponding to a response action -->
5337 <tp:SimplePart
5338   tp:id="CompanyA_Response"
5339   tp:mimetype="application/xml">
5340   <tp:NamespaceSupported
5341     tp:location="http://www.rosettanet.org/schemas/PurchaseOrderConfirmation.xsd"
5342     tp:version="1.0">
5343     http://www.rosettanet.org/schemas/PIP3A4PurchaseOrderConfirmation.xsd
5344   </tp:NamespaceSupported>
5345 </tp:SimplePart>
5346 <tp:SimplePart
5347   tp:id="CompanyB_Response"
5348   tp:mimetype="application/xml">
5349   <tp:NamespaceSupported
5350     tp:location="http://www.rosettanet.org/schemas/PurchaseOrderConfirmation.xsd"
5351     tp:version="1.0">
5352     http://www.rosettanet.org/schemas/PIP3A4PurchaseOrderConfirmation.xsd
5353   </tp:NamespaceSupported>
5354 </tp:SimplePart>
5355 <!-- An ebXML message with a SOAP Envelope only -->
5356 <tp:Packaging
5357   tp:id="CompanyA_MshSignalPackage">
5358   <tp:ProcessingCapabilities
5359     tp:parse="true"
5360     tp:generate="true"/>
5361   <tp:CompositeList>
5362     <tp:Composite
5363       tp:id="CompanyA_MshSignal"
5364       tp:mimetype="multipart/related"
5365       tp:mimeparameters="type=text/xml">
5366       <tp:Constituent tp:idref="CompanyA_MsgHdr"/>
5367     </tp:Composite>
5368   </tp:CompositeList>
5369 </tp:Packaging>
5370 <tp:Packaging
5371   tp:id="CompanyB_MshSignalPackage">
5372   <tp:ProcessingCapabilities
5373     tp:parse="true"
5374     tp:generate="true"/>
5375   <tp:CompositeList>
5376     <tp:Composite
5377       tp:id="CompanyB_MshSignal"
5378       tp:mimetype="multipart/related"
5379       tp:mimeparameters="type=text/xml">
5380       <tp:Constituent tp:idref="CompanyB_MsgHdr"/>
5381     </tp:Composite>
5382   </tp:CompositeList>
5383 </tp:Packaging>
5384 <!-- An ebXML message with a SOAP Envelope plus a request action payload -->
5385 <tp:Packaging tp:id="CompanyA_RequestPackage">

```



```

5386 <tp:ProcessingCapabilities
5387   tp:parse="true"
5388   tp:generate="true" />
5389 <tp:CompositeList>
5390   <tp:Composite
5391     tp:id="CompanyA_RequestMsg"
5392     tp:mimetype="multipart/related"
5393     tp:mimeparameters="type=text/xml">
5394     <tp:Constituent tp:idref="CompanyA_MsgHdr"/>
5395     <tp:Constituent tp:idref="CompanyA_Request"/>
5396   </tp:Composite>
5397 </tp:CompositeList>
5398 </tp:Packaging>
5399 <tp:Packaging tp:id="CompanyB_RequestPackage">
5400   <tp:ProcessingCapabilities
5401     tp:parse="true"
5402     tp:generate="true" />
5403   <tp:CompositeList>
5404     <tp:Composite
5405       tp:id="CompanyB_RequestMsg"
5406       tp:mimetype="multipart/related"
5407       tp:mimeparameters="type=text/xml">
5408       <tp:Constituent tp:idref="CompanyB_MsgHdr"/>
5409       <tp:Constituent tp:idref="CompanyB_Request"/>
5410     </tp:Composite>
5411   </tp:CompositeList>
5412 </tp:Packaging>
5413 <!-- An ebXML message with a SOAP Envelope plus a response action payload -->
5414 <tp:Packaging tp:id="CompanyA_ResponsePackage">
5415   <tp:ProcessingCapabilities tp:parse="true" tp:generate="true"/>
5416   <tp:CompositeList>
5417     <tp:Composite
5418       tp:id="CompanyA_ResponseMsg"
5419       tp:mimetype="multipart/related"
5420       tp:mimeparameters="type=text/xml">
5421       <tp:Constituent tp:idref="CompanyA_MsgHdr"/>
5422       <tp:Constituent tp:idref="CompanyA_Response"/>
5423     </tp:Composite>
5424   </tp:CompositeList>
5425 </tp:Packaging>
5426 <tp:Packaging tp:id="CompanyB_ResponsePackage">
5427   <tp:ProcessingCapabilities
5428     tp:parse="true"
5429     tp:generate="true" />
5430   <tp:CompositeList>
5431     <tp:Composite
5432       tp:id="CompanyB_ResponseMsg"
5433       tp:mimetype="multipart/related"
5434       tp:mimeparameters="type=text/xml">
5435       <tp:Constituent tp:idref="CompanyB_MsgHdr"/>
5436       <tp:Constituent tp:idref="CompanyB_Response"/>
5437     </tp:Composite>
5438   </tp:CompositeList>
5439 </tp:Packaging>
5440 <!-- An ebXML message with a SOAP Envelope plus a Receipt Acknowledgment payload -->
5441 <tp:Packaging tp:id="CompanyA_ReceiptAcknowledgmentPackage">
5442   <tp:ProcessingCapabilities
5443     tp:parse="true"
5444     tp:generate="true" />
5445   <tp:CompositeList>
5446     <tp:Composite
5447       tp:id="CompanyA_ReceiptAcknowledgmentMsg"
5448       tp:mimetype="multipart/related"
5449       tp:mimeparameters="type=text/xml">
5450       <tp:Constituent tp:idref="CompanyA_MsgHdr"/>
5451       <tp:Constituent tp:idref="CompanyA_ReceiptAcknowledgment"/>
5452     </tp:Composite>
5453   </tp:CompositeList>
5454 </tp:Packaging>
5455 <tp:Packaging tp:id="CompanyB_ReceiptAcknowledgmentPackage">
5456   <tp:ProcessingCapabilities
5457     tp:parse="true"
5458     tp:generate="true" />
5459   <tp:CompositeList>
5460     <tp:Composite
5461       tp:id="CompanyB_ReceiptAcknowledgmentMsg"
5462       tp:mimetype="multipart/related"
5463       tp:mimeparameters="type=text/xml">

```

```

5464     <tp:Constituent tp:idref="CompanyB_MsgHdr"/>
5465     <tp:Constituent tp:idref="CompanyB_ReceiptAcknowledgment"/>
5466   </tp:Composite>
5467 </tp:CompositeList>
5468 </tp:Packaging>
5469 <!-- An ebXML message with a SOAP Envelope plus an Exception payload -->
5470 <tp:Packaging tp:id="CompanyA_ExceptionPackage">
5471   <tp:ProcessingCapabilities
5472     tp:parse="true"
5473     tp:generate="true"/>
5474   <tp:CompositeList>
5475     <tp:Composite
5476       tp:id="CompanyA_ExceptionMsg"
5477       tp:mimetype="multipart/related"
5478       tp:mimeparameters="type=text/xml">
5479       <tp:Constituent tp:idref="CompanyA_MsgHdr"/>
5480       <tp:Constituent tp:idref="CompanyA_Exception"/>
5481     </tp:Composite>
5482   </tp:CompositeList>
5483 </tp:Packaging>
5484 <tp:Packaging tp:id="CompanyB_ExceptionPackage">
5485   <tp:ProcessingCapabilities
5486     tp:parse="true"
5487     tp:generate="true"/>
5488   <tp:CompositeList>
5489     <tp:Composite
5490       tp:id="CompanyB_ExceptionMsg"
5491       tp:mimetype="multipart/related"
5492       tp:mimeparameters="type=text/xml">
5493       <tp:Constituent tp:idref="CompanyB_MsgHdr"/>
5494       <tp:Constituent tp:idref="CompanyB_Exception"/>
5495     </tp:Composite>
5496   </tp:CompositeList>
5497 </tp:Packaging>
5498 <!-- An ebXML message with a Receipt Acknowledgment signal, plus a business response,
5499   or an ebXML message with an Exception signal -->
5500 <tp:Packaging tp:id="CompanyA_SyncReplyPackage">
5501   <tp:ProcessingCapabilities
5502     tp:parse="true"
5503     tp:generate="true"/>
5504   <tp:CompositeList>
5505     <tp:Composite
5506       tp:id="CompanyA_SignalAndResponseMsg"
5507       tp:mimetype="multipart/related"
5508       tp:mimeparameters="type=text/xml">
5509       <tp:Constituent tp:idref="CompanyA_MsgHdr"/>
5510       <tp:Constituent tp:idref="CompanyA_ReceiptAcknowledgment"/>
5511       <tp:Constituent tp:idref="CompanyA_Response"/>
5512     </tp:Composite>
5513   </tp:CompositeList>
5514 </tp:Packaging>
5515 <tp:Packaging tp:id="CompanyB_SyncReplyPackage">
5516   <tp:ProcessingCapabilities
5517     tp:parse="true"
5518     tp:generate="true"/>
5519   <tp:CompositeList>
5520     <tp:Composite
5521       tp:id="CompanyB_SignalAndResponseMsg"
5522       tp:mimetype="multipart/related"
5523       tp:mimeparameters="type=text/xml">
5524       <tp:Constituent tp:idref="CompanyB_MsgHdr"/>
5525       <tp:Constituent tp:idref="CompanyB_ReceiptAcknowledgment"/>
5526       <tp:Constituent tp:idref="CompanyB_Response"/>
5527     </tp:Composite>
5528   </tp:CompositeList>
5529 </tp:Packaging>
5530 <tp:Comment xml:lang="en-US">buy/sell agreement between CompanyA.com and
5531 CompanyB.com</tp:Comment>
5532 </tp:CollaborationProtocolAgreement>
5533

```

5534 **Appendix C Business Process Specification Corresponding**
 5535 **to Complete CPP and CPA Definition (Non-Normative)**

5536 This Business Process Specification referenced by the CPPs and CPA in Appendix A and
 5537 Appendix B are reproduced here. This document is available as an ASCII file at:

5538 http://www.oasis-open.org/committees/ebxml-cppa/schema/bpss-example-2_0a.xml
 5539

5540 The schema to which this instance document conforms is available as an ASCII file at:

5541 <http://www.oasis-open.org/committees/ebxml-cppa/schema/ebBPSS1.04.xsd>
 5542

```

5543 <?xml version="1.0" encoding="UTF-8"?>
5544 <ProcessSpecification
5545   xmlns="http://www.ebxml.org/BusinessProcess"
5546   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
5547   xsi:schemaLocation="http://www.ebxml.org/BusinessProcess ebBPSS1.04.xsd"
5548   name="PIP3A4RequestPurchaseOrder"
5549   uuid="urn:icann:rosettanet.org:bpid:3A4$2.0"
5550   version="R02.00">
5551   <Documentation>
5552     This PIP enables a buyer to issue a purchase order and obtain a quick response
5553     from the provider that acknowledges which of the purchase order product line
5554     items are accepted, rejected, or pending.
5555   </Documentation>
5556   <!--Purchase order Request Document-->
5557   <BusinessDocument
5558     name="Purchase Order Request"
5559     nameID="Pip3A4PurchaseOrderRequest"
5560     specificationLocation="PurchaseOrderRequest.xsd">
5561     <Documentation>
5562       The document is an XSD file that specifies the rules for creating the XML
5563       document for the business action of requesting a purchase order.
5564     </Documentation>
5565   </BusinessDocument>
5566   <BusinessDocument
5567     name="Purchase Order Confirmation"
5568     nameID="Pip3A4PurchaseOrderConfirmation"
5569     specificationLocation="PurchaseOrderConfirmation.xsd">
5570     <Documentation>
5571       The document is an XSD file that specifies the rules for creating the XML
5572       document for the business action of making a purchase order confirmation.
5573     </Documentation>
5574   </BusinessDocument>
5575   <BusinessTransaction
5576     name="Request Purchase Order"
5577     nameID="RequestPurchaseOrder_BT">
5578     <RequestingBusinessActivity
5579       name="Purchase Order Request Action"
5580       nameID="PurchaseOrderRequestAction"
5581       isAuthorizationRequired="true"
5582       isIntelligibleCheckRequired="true"
5583       isNonRepudiationReceiptRequired="true"
5584       isNonRepudiationRequired="true"
5585       timeToAcknowledgeReceipt="P0Y0M0DT2H0M0S">
5586     <DocumentEnvelope
5587       businessDocument="Purchase Order Request"
5588       businessDocumentIDRef="Pip3A4PurchaseOrderRequest"
5589       isAuthenticated="persistent"
5590       isConfidential="transient"
5591       isTamperProof="persistent"/>
5592     </RequestingBusinessActivity>
5593     <RespondingBusinessActivity
5594       name="Purchase Order Confirmation Action"
5595       nameID="PurchaseOrderConfirmationAction"
5596       isAuthorizationRequired="true"
5597       isIntelligibleCheckRequired="true"
5598       isNonRepudiationRequired="true"
5599       timeToAcknowledgeReceipt="P0Y0M0DT2H0M0S">
5600     <DocumentEnvelope
5601
```

```
5602     businessDocument="Purchase Order Confirmation"
5603     businessDocumentIDRef="Pip3A4PurchaseOrderConfirmation"
5604     isAuthenticated="persistent"
5605     isConfidential="transient"
5606     isPositiveResponse="true"
5607     isTamperProof="persistent" />
5608 </RespondingBusinessActivity>
5609 </BusinessTransaction>
5610 <BinaryCollaboration
5611   name="Request Purchase Order"
5612   nameID="RequestPurchaseOrder_BC"
5613   initiatingRole="BuyerId">
5614   <Role
5615     name="Buyer"
5616     nameID="BuyerId" />
5617   <Role
5618     name="Seller"
5619     nameID="SellerId" />
5620   <Start toBusinessState="Request Purchase Order" />
5621   <BusinessTransactionActivity
5622     name="Request Purchase Order"
5623     nameID="RequestPurchaseOrder_BTA"
5624     businessTransaction="Request Purchase Order"
5625     businessTransactionIDRef="RequestPurchaseOrder_BT"
5626     fromRole="Buyer" fromRoleIDRef="BuyerId"
5627     toRole="Seller" toRoleIDRef="SellerId"
5628     isLegallyBinding="true"
5629     timeToPerform="P0Y0M0DT24H0M0S"
5630     isConcurrent="false" />
5631   <Success
5632     fromBusinessState="Request Purchase Order"
5633     conditionGuard="Success" />
5634   <Failure
5635     fromBusinessState="Request Purchase Order"
5636     conditionGuard="BusinessFailure" />
5637   <Transition
5638     fromBusinessState="Request Purchase Order"
5639     toBusinessState="Request Purchase Order" />
5640 </BinaryCollaboration>
5641 </ProcessSpecification>
5642
5643
5644
5645
```

5646 **Appendix D W3C XML Schema Document Corresponding to**
5647 **Complete CPP and CPA Definition (Normative)**

5648 This XML Schema document is available as an ASCII file at:

5649 http://www.oasis-open.org/committees/ebxml-cppa/schema/cpp-cpa-2_0b.xsd

```
5650
5651 <?xml version="1.0" encoding="UTF-8"?>
5652 <!-- This is the schema that corresponds to the version 2.0 CPP/A spec -->
5653 <!-- Some parsers may require explicit declaration of
5654 'xmlns:xml="http://www.w3.org/XML/1998/namespace"'.
5655 In that case, a copy of this schema augmented with the above declaration should be cached
5656 and used
5657 for the purpose of schema validation for CPPs and CPAs. -->
5658 <schema
5659   targetNamespace="http://www.oasis-open.org/committees/ebxml-cppa/schema/cpp-cpa-2_0.xsd"
5660   xmlns:tns="http://www.oasis-open.org/committees/ebxml-cppa/schema/cpp-cpa-2_0.xsd"
5661   xmlns="http://www.w3.org/2001/XMLSchema"
5662   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
5663   xmlns:xlink="http://www.w3.org/1999/xlink"
5664   xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
5665   elementFormDefault="qualified"
5666   attributeFormDefault="qualified" version="2_0b">
5667   <import
5668     namespace="http://www.w3.org/1999/xlink"
5669     schemaLocation="http://www.oasis-open.org/committees/ebxml-msg/schema/xlink.xsd"/>
5670   <import
5671     namespace="http://www.w3.org/2000/09/xmldsig#"
5672     schemaLocation="http://www.w3.org/TR/xmldsig-core/xmldsig-core-schema.xsd"/>
5673   <import
5674     namespace="http://www.w3.org/XML/1998/namespace"
5675     schemaLocation="http://www.w3.org/2001/03/xml.xsd"/>
5676   <attributeGroup name="pkg.grp">
5677     <attribute ref="tns:id" use="required"/>
5678     <attribute name="mimetype" type="tns:non-empty-string" use="required"/>
5679     <attribute name="mimeparameters" type="tns:non-empty-string"/>
5680   </attributeGroup>
5681   <attributeGroup name="xlink.grp">
5682     <attribute ref="xlink:type" fixed="simple"/>
5683     <attribute ref="xlink:href" use="required"/>
5684   </attributeGroup>
5685   <element name="CollaborationProtocolAgreement">
5686     <complexType>
5687       <sequence>
5688         <element ref="tns:Status"/>
5689         <element ref="tns:Start"/>
5690         <element ref="tns:End"/>
5691         <element ref="tns:ConversationConstraints" minOccurs="0"/>
5692         <element ref="tns:PartyInfo" minOccurs="2" maxOccurs="2"/>
5693         <element ref="tns:SimplePart" maxOccurs="unbounded"/>
5694         <element ref="tns:Packaging" maxOccurs="unbounded"/>
5695         <element ref="tns:Signature" minOccurs="0"/>
5696         <element ref="tns:Comment" minOccurs="0" maxOccurs="unbounded"/>
5697       </sequence>
5698       <attribute name="cpaid" type="tns:non-empty-string" use="required"/>
5699       <attribute ref="tns:version" use="required"/>
5700     </complexType>
5701   </element>
5702   <element name="Signature">
5703     <complexType>
5704       <sequence>
5705         <element ref="ds:Signature" maxOccurs="3"/>
5706       </sequence>
5707     </complexType>
5708   </element>
5709   <element name="CollaborationProtocolProfile">
5710     <complexType>
5711       <sequence>
5712         <element ref="tns:PartyInfo" maxOccurs="unbounded"/>
5713         <element ref="tns:SimplePart" maxOccurs="unbounded"/>
5714         <element ref="tns:Packaging" maxOccurs="unbounded"/>
5715         <element ref="tns:Signature" minOccurs="0"/>
5716         <element ref="tns:Comment" minOccurs="0" maxOccurs="unbounded"/>
5717
```

```

5717     </sequence>
5718     <attribute name="cppid" type="tns:non-empty-string" use="required"/>
5719     <attribute ref="tns:version" use="required"/>
5720 </complexType>
5721 </element>
5722 <element name="ProcessSpecification">
5723   <complexType>
5724     <sequence>
5725       <element ref="ds:Reference" minOccurs="0" maxOccurs="unbounded"/>
5726     </sequence>
5727     <attribute name="name" type="tns:non-empty-string" use="required"/>
5728     <attribute ref="tns:version" use="required"/>
5729     <attributeGroup ref="tns:xlink.grp"/>
5730     <attribute name="uuid" type="anyURI"/>
5731   </complexType>
5732 </element>
5733 <element name="Service" type="tns:service.type"/>
5734 <element name="Protocol" type="tns:protocol.type"/>
5735 <element name="SendingProtocol" type="tns:protocol.type"/>
5736 <element name="ReceivingProtocol" type="tns:protocol.type"/>
5737 <element name="OverrideMshActionBinding">
5738   <complexType>
5739     <attribute name="action" type="tns:non-empty-string" use="required"/>
5740     <attribute name="channelId" type="IDREF" use="required"/>
5741   </complexType>
5742 </element>
5743 <element name="ChannelId" type="IDREF"/>
5744 <complexType name="ActionBinding.type">
5745   <sequence>
5746     <element ref="tns:BusinessTransactionCharacteristics"/>
5747     <element ref="tns:ActionContext" minOccurs="0"/>
5748     <element ref="tns:ChannelId" maxOccurs="unbounded"/>
5749     <any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
5750   </sequence>
5751   <attribute name="id" type="ID" use="required"/>
5752   <attribute name="action" type="tns:non-empty-string" use="required"/>
5753   <attribute name="packageId" type="IDREF" use="required"/>
5754   <attribute ref="xlink:href" use="optional"/>
5755   <attribute ref="xlink:type" fixed="simple"/>
5756 </complexType>
5757 <element name="ActionContext">
5758   <complexType>
5759     <sequence>
5760       <element ref="tns:CollaborationActivity" minOccurs="0"/>
5761       <any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
5762     </sequence>
5763     <attribute name="binaryCollaboration" type="tns:non-empty-string" use="required"/>
5764     <attribute name="businessTransactionActivity" type="tns:non-empty-string" use="required"/>
5765     <attribute name="requestOrResponseAction" type="tns:non-empty-string" use="required"/>
5766   </complexType>
5767 </element>
5768 <element name="CollaborationActivity">
5769   <complexType>
5770     <sequence>
5771       <element ref="tns:CollaborationActivity" minOccurs="0"/>
5772     </sequence>
5773     <attribute name="name" type="tns:non-empty-string"/>
5774   </complexType>
5775 </element>
5776 <element name="CollaborationRole">
5777   <complexType>
5778     <sequence>
5779       <element ref="tns:ProcessSpecification"/>
5780       <element ref="tns:Role"/>
5781       <element name="ApplicationCertificateRef" type="tns:CertificateRef.type" minOccurs="0"
5782 maxOccurs="unbounded"/>
5783       <element name="ApplicationSecurityDetailsRef" type="tns:SecurityDetailsRef.type"
5784 minOccurs="0"/>
5785       <element ref="tns:ServiceBinding"/>
5786     </sequence>
5787   </complexType>
5788 </element>
5789 <element name="PartyInfo">
5790   <complexType>
5791     <sequence>
5792       <element ref="tns:PartyId" maxOccurs="unbounded"/>
5793       <element ref="tns:PartyRef" maxOccurs="unbounded"/>
5794       <element ref="tns:CollaborationRole" maxOccurs="unbounded"/>

```

```

5795     <element ref="tns:Certificate" minOccurs="0" maxOccurs="unbounded"/>
5796     <element ref="tns:SecurityDetails" minOccurs="0" maxOccurs="unbounded"/>
5797     <element ref="tns:DeliveryChannel" maxOccurs="unbounded"/>
5798     <element ref="tns:Transport" maxOccurs="unbounded"/>
5799     <element ref="tns:DocExchange" maxOccurs="unbounded"/>
5800     <element ref="tns:OverrideMshActionBinding" minOccurs="0" maxOccurs="unbounded"/>
5801   </sequence>
5802   <attribute name="partyName" type="tns:non-empty-string" use="required"/>
5803   <attribute name="defaultMshChannelId" type="IDREF" use="required"/>
5804   <attribute name="defaultMshPackageId" type="IDREF" use="required"/>
5805 </complexType>
5806 </element>
5807 <element name="PartyId">
5808   <complexType>
5809     <simpleContent>
5810       <extension base="tns:non-empty-string">
5811         <attribute name="type" type="anyURI"/>
5812       </extension>
5813     </simpleContent>
5814   </complexType>
5815 </element>
5816 <element name="PartyRef">
5817   <complexType>
5818     <sequence>
5819     </sequence>
5820     <attributeGroup ref="tns:xlink.grp"/>
5821     <attribute name="type" type="anyURI"/>
5822     <attribute name="schemaLocation" type="anyURI"/>
5823   </complexType>
5824 </element>
5825 <element name="DeliveryChannel">
5826   <complexType>
5827     <sequence>
5828       <element ref="tns:MessagingCharacteristics"/>
5829     </sequence>
5830     <attribute name="channelId" type="ID" use="required"/>
5831     <attribute name="transportId" type="IDREF" use="required"/>
5832     <attribute name="docExchangeId" type="IDREF" use="required"/>
5833   </complexType>
5834 </element>
5835 <element name="Transport">
5836   <complexType>
5837     <sequence>
5838       <element ref="tns:TransportSender" minOccurs="0"/>
5839       <element ref="tns:TransportReceiver" minOccurs="0"/>
5840     </sequence>
5841     <attribute name="transportId" type="ID" use="required"/>
5842   </complexType>
5843 </element>
5844 <element name="AccessAuthentication" type="tns:accessAuthentication.type"/>
5845 <element name="TransportSender">
5846   <complexType>
5847     <sequence>
5848       <element name="TransportProtocol" type="tns:protocol.type"/>
5849       <element ref="tns:AccessAuthentication" minOccurs="0" maxOccurs="unbounded"/>
5850       <element ref="tns:TransportClientSecurity" minOccurs="0"/>
5851     </sequence>
5852   </complexType>
5853 </element>
5854 <element name="TransportReceiver">
5855   <complexType>
5856     <sequence>
5857       <element name="TransportProtocol" type="tns:protocol.type"/>
5858       <element ref="tns:AccessAuthentication" minOccurs="0" maxOccurs="unbounded"/>
5859       <element ref="tns:Endpoint" maxOccurs="unbounded"/>
5860       <element ref="tns:TransportServerSecurity" minOccurs="0"/>
5861     </sequence>
5862   </complexType>
5863 </element>
5864 <element name="Endpoint">
5865   <complexType>
5866     <attribute name="uri" type="anyURI" use="required"/>
5867     <attribute name="type" type="tns:endpointType.type" default="allPurpose"/>
5868   </complexType>
5869 </element>
5870 <element name="TransportClientSecurity">
5871   <complexType>
5872     <sequence>

```

```

5873      <element name="TransportSecurityProtocol" type="tns:protocol.type"/>
5874      <element name="ClientCertificateRef" type="tns:CertificateRef.type" minOccurs="0"/>
5875      <element name="ServerSecurityDetailsRef" type="tns:SecurityDetailsRef.type"
5876      minOccurs="0"/>
5877      <element ref="tns:EncryptionAlgorithm" minOccurs="0" maxOccurs="unbounded"/>
5878      </sequence>
5879      </complexType>
5880    </element>
5881    <element name="TransportServerSecurity">
5882      <complexType>
5883        <sequence>
5884          <element name="TransportSecurityProtocol" type="tns:protocol.type"/>
5885          <element name="ServerCertificateRef" type="tns:CertificateRef.type"/>
5886          <element name="ClientSecurityDetailsRef" type="tns:SecurityDetailsRef.type"
5887          minOccurs="0"/>
5888          <element ref="tns:EncryptionAlgorithm" minOccurs="0" maxOccurs="unbounded"/>
5889          </sequence>
5890        </complexType>
5891      </element>
5892      <element name="Certificate">
5893        <complexType>
5894          <sequence>
5895            <element ref="ds:KeyInfo"/>
5896            </sequence>
5897            <attribute name="certId" type="ID" use="required"/>
5898          </complexType>
5899        </element>
5900      <element name="DocExchange">
5901        <complexType>
5902          <sequence>
5903            <element ref="tns:ebXMLSenderBinding" minOccurs="0"/>
5904            <element ref="tns:ebXMLReceiverBinding" minOccurs="0"/>
5905            </sequence>
5906            <attribute name="docExchangeId" type="ID" use="required"/>
5907          </complexType>
5908        </element>
5909      <element name="ReliableMessaging">
5910        <complexType>
5911          <sequence>
5912            <element name="Retries" type="integer" minOccurs="0"/>
5913            <element name="RetryInterval" type="duration" minOccurs="0"/>
5914            <element name="MessageOrderSemantics" type="tns:messageOrderSemantics.type"/>
5915            </sequence>
5916          </complexType>
5917        </element>
5918      <element name="PersistDuration" type="duration"/>
5919      <element name="SenderNonRepudiation">
5920        <complexType>
5921          <sequence>
5922            <element name="NonRepudiationProtocol" type="tns:protocol.type"/>
5923            <element ref="tns:HashFunction"/>
5924            <element ref="tns:SignatureAlgorithm" maxOccurs="unbounded"/>
5925            <element name="SigningCertificateRef" type="tns:CertificateRef.type"/>
5926            </sequence>
5927          </complexType>
5928        </element>
5929      <element name="ReceiverNonRepudiation">
5930        <complexType>
5931          <sequence>
5932            <element name="NonRepudiationProtocol" type="tns:protocol.type"/>
5933            <element ref="tns:HashFunction"/>
5934            <element ref="tns:SignatureAlgorithm" maxOccurs="unbounded"/>
5935            <element name="SigningSecurityDetailsRef" type="tns:SecurityDetailsRef.type"
5936            minOccurs="0"/>
5937            </sequence>
5938          </complexType>
5939        </element>
5940      <element name="HashFunction" type="tns:non-empty-string"/>
5941      <element name="EncryptionAlgorithm">
5942        <complexType>
5943          <simpleContent>
5944            <extension base="tns:non-empty-string">
5945              <attribute name="minimumStrength" type="integer"/>
5946              <attribute name="oid" type="tns:non-empty-string"/>
5947              <attribute name="w3c" type="tns:non-empty-string"/>
5948              <attribute name="enumerationType" type="tns:non-empty-string"/>
5949            </extension>
5950          </simpleContent>

```



```

5951     </complexType>
5952 </element>
5953 <element name="SignatureAlgorithm">
5954   <complexType>
5955     <simpleContent>
5956       <extension base="tns:non-empty-string">
5957         <attribute name="oid" type="tns:non-empty-string"/>
5958         <attribute name="w3c" type="tns:non-empty-string"/>
5959         <attribute name="enumerationType" type="tns:non-empty-string"/>
5960       </extension>
5961     </simpleContent>
5962   </complexType>
5963 </element>
5964 <element name="SenderDigitalEnvelope">
5965   <complexType>
5966     <sequence>
5967       <element name="DigitalEnvelopeProtocol" type="tns:protocol.type"/>
5968       <element ref="tns:EncryptionAlgorithm" maxOccurs="unbounded"/>
5969       <element name="EncryptionSecurityDetailsRef" type="tns:SecurityDetailsRef.type"
5970 minOccurs="0"/>
5971     </sequence>
5972   </complexType>
5973 </element>
5974 <element name="ReceiverDigitalEnvelope">
5975   <complexType>
5976     <sequence>
5977       <element name="DigitalEnvelopeProtocol" type="tns:protocol.type"/>
5978       <element ref="tns:EncryptionAlgorithm" maxOccurs="unbounded"/>
5979       <element name="EncryptionCertificateRef" type="tns:CertificateRef.type"/>
5980     </sequence>
5981   </complexType>
5982 </element>
5983 <element name="ebXMLSenderBinding">
5984   <complexType>
5985     <sequence>
5986       <element ref="tns:ReliableMessaging" minOccurs="0"/>
5987       <element ref="tns:PersistDuration" minOccurs="0"/>
5988       <element ref="tns:SenderNonRepudiation" minOccurs="0"/>
5989       <element ref="tns:SenderDigitalEnvelope" minOccurs="0"/>
5990       <element ref="tns:NamespaceSupported" minOccurs="0" maxOccurs="unbounded"/>
5991     </sequence>
5992     <attribute ref="tns:version" use="required"/>
5993   </complexType>
5994 </element>
5995 <element name="ebXMLReceiverBinding">
5996   <complexType>
5997     <sequence>
5998       <element ref="tns:ReliableMessaging" minOccurs="0"/>
5999       <element ref="tns:PersistDuration" minOccurs="0"/>
6000       <element ref="tns:ReceiverNonRepudiation" minOccurs="0"/>
6001       <element ref="tns:ReceiverDigitalEnvelope" minOccurs="0"/>
6002       <element ref="tns:NamespaceSupported" minOccurs="0" maxOccurs="unbounded"/>
6003     </sequence>
6004     <attribute ref="tns:version" use="required"/>
6005   </complexType>
6006 </element>
6007 <element name="NamespaceSupported">
6008   <complexType>
6009     <simpleContent>
6010       <extension base="anyURI">
6011         <attribute name="location" type="anyURI" use="required"/>
6012         <attribute ref="tns:version"/>
6013       </extension>
6014     </simpleContent>
6015   </complexType>
6016 </element>
6017 <element name="BusinessTransactionCharacteristics">
6018   <complexType>
6019     <attribute name="isNonRepudiationRequired" type="boolean"/>
6020     <attribute name="isNonRepudiationReceiptRequired" type="boolean"/>
6021     <attribute name="isConfidential" type="tns:persistenceLevel.type"/>
6022     <attribute name="isAuthenticated" type="tns:persistenceLevel.type"/>
6023     <attribute name="isTamperProof" type="tns:persistenceLevel.type"/>
6024     <attribute name="isAuthorizationRequired" type="boolean"/>
6025     <attribute name="isIntelligibleCheckRequired" type="boolean"/>
6026     <attribute name="timeToAcknowledgeReceipt" type="duration"/>
6027     <attribute name="timeToAcknowledgeAcceptance" type="duration"/>
6028     <attribute name="timeToPerform" type="duration"/>

```

```

6029     <attribute name="retryCount" type="integer"/>
6030   </complexType>
6031 </element>
6032 <element name="MessagingCharacteristics">
6033   <complexType>
6034     <attribute ref="tns:syncReplyMode" default="none"/>
6035     <attribute name="ackRequested" type="tns:perMessageCharacteristics.type"
6036 default="perMessage"/>
6037     <attribute name="ackSignatureRequested" type="tns:perMessageCharacteristics.type"
6038 default="perMessage"/>
6039     <attribute name="duplicateElimination" type="tns:perMessageCharacteristics.type"
6040 default="perMessage"/>
6041     <attribute name="actor" type="tns:actor.type"/>
6042   </complexType>
6043 </element>
6044 <element name="ServiceBinding">
6045   <complexType>
6046     <sequence>
6047       <element ref="tns:Service"/>
6048       <element ref="tns:CanSend" minOccurs="0" maxOccurs="unbounded"/>
6049       <element ref="tns:CanReceive" minOccurs="0" maxOccurs="unbounded"/>
6050     </sequence>
6051   </complexType>
6052 </element>
6053 <element name="CanSend">
6054   <complexType>
6055     <sequence>
6056       <element name="ThisPartyActionBinding" type="tns:ActionBinding.type"/>
6057       <element name="OtherPartyActionBinding" type="IDREF" minOccurs="0"/>
6058       <element ref="tns:CanReceive" minOccurs="0" maxOccurs="unbounded"/>
6059     </sequence>
6060   </complexType>
6061 </element>
6062 <element name="CanReceive">
6063   <complexType>
6064     <sequence>
6065       <element name="ThisPartyActionBinding" type="tns:ActionBinding.type"/>
6066       <element name="OtherPartyActionBinding" type="IDREF" minOccurs="0"/>
6067       <element ref="tns:CanSend" minOccurs="0" maxOccurs="unbounded"/>
6068     </sequence>
6069   </complexType>
6070 </element>
6071 <element name="Status">
6072   <complexType>
6073     <attribute name="value" type="tns:statusValue.type" use="required"/>
6074   </complexType>
6075 </element>
6076 <element name="Start" type="dateTime"/>
6077 <element name="End" type="dateTime"/>
6078 <element name="Type" type="tns:non-empty-string"/>
6079 <element name="ConversationConstraints">
6080   <complexType>
6081     <attribute name="invocationLimit" type="int"/>
6082     <attribute name="concurrentConversations" type="int"/>
6083   </complexType>
6084 </element>
6085 <element name="Role">
6086   <complexType>
6087     <attribute name="name" type="tns:non-empty-string" use="required"/>
6088     <attributeGroup ref="tns:xlink.grp"/>
6089   </complexType>
6090 </element>
6091 <element name="SignatureTransforms">
6092   <complexType>
6093     <sequence>
6094       <element ref="ds:Transform" maxOccurs="unbounded"/>
6095     </sequence>
6096   </complexType>
6097 </element>
6098 <element name="EncryptionTransforms">
6099   <complexType>
6100     <sequence>
6101       <element ref="ds:Transform" maxOccurs="unbounded"/>
6102     </sequence>
6103   </complexType>
6104 </element>
6105 <element name="Constituent">
6106   <complexType>

```

```

6107
6108
6109
6110
6111
6112
6113
6114
6115
6116
6117
6118
6119
6120
6121
6122
6123
6124
6125
6126
6127
6128
6129
6130
6131
6132
6133
6134
6135
6136
6137
6138
6139
6140
6141
6142
6143
6144
6145
6146
6147
6148
6149
6150
6151
6152
6153
6154
6155
6156
6157
6158
6159
6160
6161
6162
6163
6164
6165
6166
6167
6168
6169
6170
6171
6172
6173
6174
6175
6176
6177
6178
6179
6180
6181
6182
6183
6184

<sequence>
  <element ref="tns:SignatureTransforms" minOccurs="0"/>
  <element ref="tns:EncryptionTransforms" minOccurs="0"/>
</sequence>
<attribute ref="tns:idref" use="required"/>
<attribute name="excludedFromSignature" type="boolean" default="false"/>
<attribute name="minOccurs" type="nonNegativeInteger"/>
<attribute name="maxOccurs" type="nonNegativeInteger"/>
</complexType>
</element>
<element name="Packaging">
  <complexType>
    <sequence>
      <element name="ProcessingCapabilities">
        <complexType>
          <attribute name="parse" type="boolean" use="required"/>
          <attribute name="generate" type="boolean" use="required"/>
        </complexType>
      </element>
      <element name="CompositeList" maxOccurs="unbounded">
        <complexType>
          <choice maxOccurs="unbounded">
            <element name="Encapsulation">
              <complexType>
                <sequence>
                  <element ref="tns:Constituent"/>
                </sequence>
                <attributeGroup ref="tns:pkg.grp"/>
              </complexType>
            </element>
            <element name="Composite">
              <complexType>
                <sequence>
                  <element ref="tns:Constituent" maxOccurs="unbounded"/>
                </sequence>
                <attributeGroup ref="tns:pkg.grp"/>
              </complexType>
            </element>
          </choice>
        </complexType>
      </element>
    </sequence>
    <attribute ref="tns:id" use="required"/>
  </complexType>
</element>
<element name="Comment">
  <complexType>
    <simpleContent>
      <extension base="tns:non-empty-string">
        <attribute ref="xml:lang"/>
      </extension>
    </simpleContent>
  </complexType>
</element>
<element name="SimplePart">
  <complexType>
    <sequence>
      <element ref="tns:NamespaceSupported" minOccurs="0" maxOccurs="unbounded"/>
    </sequence>
    <attributeGroup ref="tns:pkg.grp"/>
    <attribute ref="xlink:role"/>
  </complexType>
</element>
<!-- COMMON -->
<simpleType name="statusValue.type">
  <restriction base="NMTOKEN">
    <enumeration value="agreed"/>
    <enumeration value="signed"/>
    <enumeration value="proposed"/>
  </restriction>
</simpleType>
<simpleType name="endpointType.type">
  <restriction base="NMTOKEN">
    <enumeration value="login"/>
    <enumeration value="request"/>
    <enumeration value="response"/>
    <enumeration value="error"/>
    <enumeration value="allPurpose"/>
  </restriction>
</simpleType>

```

```

6185     </restriction>
6186 </simpleType>
6187 <simpleType name="non-empty-string">
6188   <restriction base="string">
6189     <minLength value="1"/>
6190   </restriction>
6191 </simpleType>
6192 <simpleType name="syncReplyMode.type">
6193   <restriction base="NMTOKEN">
6194     <enumeration value="mshSignalsOnly"/>
6195     <enumeration value="responseOnly"/>
6196     <enumeration value="signalsAndResponse"/>
6197     <enumeration value="signalsOnly"/>
6198     <enumeration value="none"/>
6199   </restriction>
6200 </simpleType>
6201 <complexType name="service.type">
6202   <simpleContent>
6203     <extension base="tns:non-empty-string">
6204       <attribute name="type" type="tns:non-empty-string"/>
6205     </extension>
6206   </simpleContent>
6207 </complexType>
6208 <complexType name="protocol.type">
6209   <simpleContent>
6210     <extension base="tns:non-empty-string">
6211       <attribute ref="tns:version"/>
6212     </extension>
6213   </simpleContent>
6214 </complexType>
6215 <attribute name="idref" type="IDREF"/>
6216 <attribute name="id" type="ID"/>
6217 <attribute name="version" type="tns:non-empty-string"/>
6218 <attribute name="syncReplyMode" type="tns:syncReplyMode.type"/>
6219 <complexType name="SecurityPolicy.type">
6220 <complexType name="CertificateRef.type">
6221   <attribute name="certId" type="IDREF" use="required"/>
6222 </complexType>
6223 <simpleType name="perMessageCharacteristics.type">
6224   <restriction base="NMTOKEN">
6225     <enumeration value="always"/>
6226     <enumeration value="never"/>
6227     <enumeration value="perMessage"/>
6228   </restriction>
6229 </simpleType>
6230 <simpleType name="actor.type">
6231   <restriction base="NMTOKEN">
6232     <enumeration value="urn:oasis:names:tc:ebxml-msg:actor:nextMSH"/>
6233     <enumeration value="urn:oasis:names:tc:ebxml-msg:actor:toPartyMSH"/>
6234   </restriction>
6235 </simpleType>
6236 <simpleType name="messageOrderSemantics.type">
6237   <restriction base="Name">
6238     <enumeration value="Guaranteed"/>
6239     <enumeration value="NotGuaranteed"/>
6240   </restriction>
6241 </simpleType>
6242 <complexType name="SecurityDetailsRef.type">
6243   <attribute name="securityId" type="IDREF" use="required"/>
6244 </complexType>
6245 <simpleType name="persistenceLevel.type">
6246   <restriction base="Name">
6247     <enumeration value="none"/>
6248     <enumeration value="transient"/>
6249     <enumeration value="persistent"/>
6250     <enumeration value="transient-and-persistent"/>
6251   </restriction>
6252 </simpleType>
6253 <element name="SecurityDetailsRef" type="tns:SecurityDetailsRef.type"/>
6254 <element name="SecurityDetails">
6255   <complexType>
6256     <sequence>
6257       <element ref="tns:TrustAnchors" minOccurs="0"/>
6258       <element ref="tns:SecurityPolicy" minOccurs="0"/>
6259     </sequence>
6260     <attribute name="securityId" type="ID" use="required"/>
6261   </complexType>
6262 </element>

```

```
6263 <element name="TrustAnchors">
6264 <complexType>
6265 <sequence>
6266 <element name="AnchorCertificateRef" type="tns:CertificateRef.type"
6267 maxOccurs="unbounded" />
6268 </sequence>
6269 </complexType>
6270 </element>
6271 <element name="SecurityPolicy">
6272 <complexType>
6273 <sequence>
6274 </sequence>
6275 </complexType>
6276 </element>
6277 <simpleType name="accessAuthentication.type">
6278 <restriction base="NMTOKEN">
6279 <enumeration value="basic" />
6280 <enumeration value="digest" />
6281 </restriction>
6282 </simpleType>
6283 </schema>
6284
```

6285 **Appendix E CPA Composition (Non-Normative)**

6286 **E.1 Suggestions for Design of Computational Procedures**

6287 A quick inspection of the schemas for the top level elements, *CollaborationProtocolProfile*
6288 (*CPP*) and *CollaborationProtocolAgreement* (*CPA*), shows that a *CPA* can be viewed as a
6289 result of merging portions of the *PartyInfo* elements found in constituent *CPPs*, and then
6290 integrating these *PartyInfo* elements with other *CPA* sibling elements, such as those governing
6291 the *CPA* validity period.

6292
6293 Merging *CPPs* into *CPAs* is one way in which trading partners can arrive at a proposed or
6294 “draft” *CPA*. A draft *CPA* might also be formed from a *CPA* template. A *CPA*-template
6295 represents one party’s proposed implementation of a business process that uses placeholder
6296 values for the identifying aspects of the other party, such as *PartyId* or *TransportEndpoint*
6297 elements. To form a *CPA* from a *CPA* template, the placeholder values are replaced by the actual
6298 values for the other trading partner. The actual values could themselves be extracted from the
6299 other trading partner’s *CPP*, if one is available, or they could be obtained from an administrator
6300 performing data entry functions.

6301
6302 We call objects draft *CPAs* to indicate their potential use as inputs to a *CPA* negotiation process
6303 in which a draft *CPA* is verified as suitable for both *Parties*, modified until a suitable *CPA* is
6304 found, or discovered to not be feasible until one side (or both) acquires additional software
6305 capabilities. These negotiation procedures and protocols are currently being designed, their
6306 requirements having been defined, and the resulting specifications should be available with the
6307 next release of this specification. In general, a draft *CPA* will constitute a proposal about an
6308 overall binding of a business process to a delivery implementation, while negotiation will be
6309 used to arrive at detailed values for parameters reflecting a final agreement. A special companion
6310 document, the *NegotiationDescriptorDocument*, provides both focus on what parameters can be
6311 negotiated as well as ranges or sets of acceptable values for those parameters.

6312
6313 In the remainder of this appendix, the goal will be to identify and describe the basic tasks that
6314 computational procedures for the assembly of the draft *CPA* would normally accomplish. While
6315 no normative specification is provided for an algorithm for *CPA* formation, some guidance for
6316 implementers is provided. This information might assist the software implementer in designing a
6317 partially automated and partially interactive software system useful for configuring *Business*
6318 *Collaboration* so as to arrive at satisfactorily complete levels of interoperability.

6319
6320 Before enumerating and describing the basic tasks, it is worthwhile mentioning two basic reasons
6321 why we focus on the component tasks involved in *CPA* formation rather than attempt to provide
6322 an algorithm for *CPA* formation. These reasons provide some hints to implementers about ways
6323 in which they might customize their approaches to drafting *CPAs* from *CPPs*.

6324 6325 **E.1.1 Variability in Inputs**

6326 User preferences provide one source of variability in the inputs to the *CPA* formation process.
6327 Let us suppose in this section that each of the *Parties* has made its *CPP* available to potential

6328 collaborators. Normally one *Party* will have a desired *Business Collaboration* (defined in a
6329 ***ProcessSpecification*** document) to implement with its intended collaborator. So the information
6330 inputs will normally involve a user preference about intended *Business Collaborations* in
6331 addition to just the *CPPs*.
6332

6333 A *CPA* formation tool can have access to local user information not advertised in the *CPP* that
6334 can contribute to the *CPA* that is formed. A user can have chosen to only advertise those system
6335 capabilities that reflect capabilities that have not been deprecated. For example, a user can only
6336 advertise HTTP and omit FTP, even when capable of using FTP. The reason for omitting FTP
6337 might be concerns about the scalability of managing user accounts, directories, and passwords
6338 for FTP sessions. Despite not advertising an FTP capability, configuration software can use tacit
6339 knowledge about its own FTP capability to form a *CPA* with an intended collaborator who
6340 happens to have only an FTP capability for implementing a desired *Business Collaboration*. In
6341 other words, business interests can, in this case, override the deprecation policy. Both tacit
6342 knowledge and detailed preference information account for variability in inputs into the *CPA*
6343 formation process.
6344

6345 **E.1.2 Variable Stringency in Evaluating Proposed Agreements**

6346 The conditions for output of a *CPA* given two *CPPs* can involve different levels and extents of
6347 interoperability. In other words, when an optimal solution that satisfies every level of
6348 requirement and every other additional constraint does not exist, a *Party* can propose a *CPA* that
6349 satisfies enough of the requirements for “a good enough” implementation. User input can be
6350 solicited to determine what is a good enough implementation, and so can be as varied as there
6351 are user configuration options to express preferences. In practice, compromises can be made on
6352 security, reliable messaging, levels of signals and acknowledgments, and other matters in order
6353 to find some acceptable means of doing business.
6354

6355 A *CPA* can support a fully interoperable configuration in which agreement has been reached on
6356 all technical levels needed for *Business Collaboration*. In such a case, matches in capabilities
6357 will have been found in all relevant technical levels.
6358

6359 However, there can be interoperable configurations agreed to in a *CPA* in which not all aspects
6360 of a *Business Collaboration* match. Gaps can exist in packaging, security, signaling, reliable
6361 messaging and other areas and yet the systems can still transport the business data, and special
6362 means can be employed to handle the exceptions. In such situations, a *CPA* can reflect
6363 configured policies or expressly solicited user permission to ignore some shortcomings in
6364 configurations. A system might not be capable of responding in a *Business Collaboration* so as
6365 to support a specified ability to supply non-repudiation of receipt, but might still be acceptable
6366 for business reasons. A system might not be able to handle all the processing needed to support,
6367 for example, SOAP with Attachments and yet still be able to treat the multipart according to
6368 "multipart/mixed" handling and allow *Business Collaboration* to take place. In fact, short of a
6369 failure to be able to transport data and a failure to be able to provide data relevant to the *Business*
6370 *Collaboration*, there are few features that might not be temporarily or indefinitely compromised
6371 about, given overriding *business* interests. This situation of "partial interoperability" is to be

6372 expected to persist for some time, and so interferes with formulating a "clean" algorithm for
6373 deciding on what is sufficient for interoperability.
6374

6375 **E.2 CPA Formation Component Tasks**

6376 Technically viewed, a *CPA* provides "bindings" between *Business Collaboration* specifications
6377 (such as those defined within the *ProcessSpecification*'s referenced documents) and those
6378 services and protocols that are used to implement these specifications. The implementation takes
6379 place at several levels and involves varied services at these levels. A *CPA* that arrives at a fully
6380 interoperable collaboration binding can be thought of as arriving at interoperable, application-to-
6381 application integration. *CPAs* can fall short of this goal and still be both useful and acceptable to
6382 the collaborating *Parties*. Certainly, if no matching data-transport capabilities can be discovered,
6383 a *CPA* would not provide much in the way of interoperable integration. Likewise, partial *CPAs*
6384 can leave significant system work to be done before a completely satisfactory application-to-
6385 application integration is realized. Even so, partial integration can be sufficient to allow
6386 collaboration, and to enjoy payoffs from increased levels of automation.
6387

6388 In practice, the *CPA* formation process can produce a complete *CPA*, a failure result, a gap list
6389 that drives a dialog with the user, or perhaps even a *CPA* that implements partial interoperability
6390 "good enough" for the business collaborators. Because both matching capabilities and
6391 interoperability can be matters of degree, the constituent tasks are finding the matches in
6392 capabilities at different levels and for different services. We next proceed to characterize the
6393 most important of these constituent tasks.
6394

6395 **E.3 CPA Formation from CPPs: Context of Tasks**

6396 To simplify discussion, assume in the following that we are viewing the tasks faced by a
6397 software agent when:
6398

- 6399 1. an intended collaborator is known and the collaborator's *CPP* has been retrieved,
- 6400 2. the *ProcessSpecification* between our side and our intended collaborator has been
6401 selected,
- 6402 3. the *Service*, *Action*, and the specific *Role* elements that our software agent is to play in
6403 the *Business Collaboration* (with discussion soon restricted to *BinaryCollaborations*) is
6404 known, and
- 6405 4. finally, the capabilities that we have advertised in our *CPP* are known
6406

6407 For vividness, we will develop our discussions using the "3A4" ebBPSS example and the *CPPs*
6408 of Company A and B that are found in full in appendices of this document and that should also
6409 be available at the web site for the OASIS ebXML CPPA Technical Committee. For simplicity,
6410 we will assume that the information about capabilities is restricted to what is available in our
6411 agent's *CPP*, and in the *CPP* of our intended collaborator. We will suppose that we have taken
6412 on the viewpoint of Company A assembling a draft *CPA*. Please note that there is no guarantee
6413 that the same draft *CPAs* will be produced in the same order from differing viewpoints.
6414

6415 In general, the basic tasks consist of finding "matches" between our capabilities and our intended
6416 collaborator's capabilities at the various levels of the collaboration protocol stack and with
6417 respect to the services supplied at these various levels. This stack, which need not be
6418 characterized in any detail, is at least distinguished by an application level and a messaging
6419 transfer level. The application level is governed by a business process flow specification, such as
6420 ebBPSS. The messaging transfer level will consist of a number of requirements and options
6421 concerning transfer protocols, security, packaging, and messaging patterns (such as various kinds
6422 of acknowledgment, error messages, and the like.)

6423
6424 In actually assembling the tasks into a computational process, it will generally make sense to
6425 perform the tasks in a certain order. The overall order reflects the implicit structure of the *CPA*:
6426 first undertake those tasks to ensure that there is a match with respect to the *Business*
6427 *Collaboration* process. Without finding that the collaborators can participate in the same
6428 **ProcessSpecification** successfully, there is little point in working through implementation
6429 options. Then, examine the matches within the components of the bindings that have been
6430 announced for the *Business Collaboration* process, checking for the most indispensable
6431 "matches" first (**Transport**-related), and continuing checks on the other layers reflecting
6432 integrated interoperability at packaging, security, signals and protocol patterns, and so on. With
6433 this basic overview in mind, let us proceed to consider the basic tasks in greater detail.

6434

6435 **E.4 Business Collaboration Process Matching Tasks**

6436 Company A has announced within its *CPP*, at least one **PartyInfo** element. For current purposes,
6437 the most important initial focus is on all the sibling elements with the path
6438 **/CollaborationProtocolProfile/PartyInfo/CollaborationRole**. Each element of this kind has a
6439 child, **ProcessSpecification**. Our initial matching task (probably better viewed as a filtering
6440 task) is to select those nodes where the **ProcessSpecification** is one that we are interested in
6441 building a CPA for! Checking the attribute values allows us to select by comparing values in the
6442 **name**, **xlink:href** or **uuid** attributes. The definitive value for matching ebBPSS process
6443 specifications is the value found in the **ProcessSpecification/@uuid** attribute.

6444

6445 **E.4.1 Matching ProcessSpecification/Roles, and Actions: Initial Filtering and Selection**

6446 The previous task has essentially found two **CollaborationRole** node sets within our and our
6447 collaborator's CPP documents where the **ProcessSpecifications** are identical, and equal to the
6448 value of interest given above. In other words, we have **CollaborationRoles** with
6449 **ProcessSpecification/@name='PIP3A4RequestPurchaseOrder'**. It is convenient but not essential
6450 to use the **name** attribute in performing this selection.

6451

6452 We next proceed to filter these node sets. We have been given our **Role** element value for our
6453 participation in the **ProcessSpecification**. For Company A, this **Role** has the **name** attribute with
6454 value 'Buyer'. Because we are here considering only **BinaryCollaborations** in ebBPSS
6455 terminology (or their equivalent in other flow languages), we are only interested in those
6456 **CollaborationRole** node sets within our collaborator's CPP that have a **Role** value equal to
6457 'Seller.' So we assume we have narrowed our focus to **CollaborationRole** node sets in Company

6458 A's *CPP* with *Role/@name='Buyer'* and in Company B's *CollaborationRole* node sets with
 6459 *Role/@name='Seller'*.

6460
 6461 For more general collaborations, such as in the *MultiPartyCollaborations* of ebBPSS, we would
 6462 need to know the list of roles available within the process, and keep track of that for each of the
 6463 *CollaborationRoles*, the *Role* values chosen correspond correctly for the participants. We do not
 6464 here discuss the matching/filtering task for collaborations involving more than two roles, as
 6465 multiparty *CPAs* are not within scope for version 2.0 of this specification.
 6466

6467 **E.5 Implementation Matching Tasks**

6468 After filtering the *CollaborationRoles* with the desired *ProcessSpecification*, we should find one
 6469 *CollaborationRole* in our own *CPP* where we play the Buyer role, and one *CollaborationRole* in
 6470 our intended collaborator Company B's *CPP* where it plays the Seller role.

6471
 6472 Our next task is to locate the specific candidate *bindings* relevant to *CPA* formation. There are
 6473 *bindings* for *Service* and *Actions*. For initial simplicity, we consider detailed matching tasks as
 6474 they arise for a standard collaboration case involving a *Request* action, followed by a *Response*
 6475 action. For version 2.0 of this specification, most matching tasks will involve matching of
 6476 referenced components of the *CPP*'s *ThisPartyActionBinding* elements under
 6477 *CollaborationRole/ServiceBinding/CanSend/* and under
 6478 *CollaborationRole/ServiceBinding/CanReceive.*

6480 **E.5.1 Action Correspondence and Selecting Correlative PackageIds, and ChannelIds**

6481 In *CPPs*, under each of the elements, *CollaborationRole/ServiceBinding/CanSend* and
 6482 *CollaborationRole/ServiceBinding/CanReceive*, are lists of *ThisPartyActionBindings*. For
 6483 *Request-Response* collaboration patterns, we are interested in matches:

- 6484
- 6485 1. in the *bindings* of the Requesting side's *CanSend/ThisPartyActionBinding* with the
 - 6486 *Responding* side's *CanReceive/ThisPartyActionBinding* for the request action, and
 - 6487 2. in the *bindings* of the Responding side's *CanSend/ThisPartyActionBinding* with the
 - 6488 *Requesting* side's *CanReceive/ThisPartyActionBinding* for the response action.

6489
 6490 These correlative *bindings* give us references to detailed components that need to match for a
 6491 fully interoperable agreement. Case 1 pertains to the *Request*. Case 2 pertains to the *Response*.

6492
 6493 For example, for Company A, we find under *CanSend*:

```
6494
6495 <tp:ThisPartyActionBinding tp:action="Purchase Order Request
6496 Action" tp:packageId="CompanyA_RequestPackage">
6497   <tp:BusinessTransactionCharacteristics ... />
6498   <tp:ActionContext tp:binaryCollaboration="Request Purchase
6499 Order" tp:businessTransactionActivity="Request Purchase
6500 Order" tp:requestOrResponseAction="Purchase Order Request
6501 Action"/>
```

```
6502     <tp:ChannelId>asyncChannelA1</tp:ChannelId>
6503 </tp:ThisPartyActionBinding>
```

6504

6505 Correlative to this, for Company B, we find under **CanReceive**:

6506

```
6507 <tp:ThisPartyActionBinding tp:action="Purchase Order Request
6508 Action" tp:packageId="CompanyB_RequestPackage">
```

```
6509     <tp:BusinessTransactionCharacteristics ... />
```

```
6510     <tp:ActionContext tp:binaryCollaboration="Request Purchase Order"
```

```
6511     tp:businessTransactionActivity="Request Purchase Order"
```

```
6512     tp:requestOrResponseAction="Purchase Order Request Action"/>
```

```
6513     <tp:ChannelId>asyncChannelB1</tp:ChannelId>
```

```
6514 </tp:ThisPartyActionBinding>
```

6515

6516 The correlation of elements can normally (when we are dealing with BPSS **Binary**
6517 **Collaborations** or their equivalents in other representations) be based on equality of the **action**
6518 (or **requestOrResponseAction**) values. More detailed correlation of elements can make use of
6519 more detailed testing and comparisons of the values in the **ActionContext** child elements of the
6520 relevant **CanSend** and **CanReceive** pairs.

6521

6522 In the preceding, we have illustrated the matching of **CanSend** and **CanReceive** for
6523 asynchronous bindings. All **CanSend** bindings that are siblings under a **ServiceBinding** element
6524 are asynchronous and make of use separate TCP connections that the **CanSend** side initiates on a
6525 listening TCP port. In order to represent binding details for synchronous sending, the convention
6526 is adopted whereby the **CanReceive** element for a Sender is placed under its **CanSend** element.
6527 This is illustrated by:

6528

```
6529 <tp:CanSend>
```

```
6530     <tp:ThisPartyActionBinding
```

```
6531     tp:id="companyA_ABID6"
```

```
6532     tp:action="Purchase Order Request Action"
```

```
6533     tp:packageId="CompanyA_RequestPackage">
```

```
6534 <tp:BusinessTransactionCharacteristics
```

```
6535     tp:isNonRepudiationRequired="true"
```

```
6536     tp:isNonRepudiationReceiptRequired="true"
```

```
6537     tp:isConfidential="transient"
```

```
6538     tp:isAuthenticated="persistent"
```

```
6539     tp:isTamperProof="persistent"
```

```
6540     tp:isAuthorizationRequired="true"
```

```
6541     tp:timeToAcknowledgeReceipt="PT2H"
```

```
6542     tp:timeToPerform="P1D"/>
```

```
6543 <tp:ActionContext
```

```
6544     tp:binaryCollaboration="Request Purchase Order"
```

```
6545     tp:businessTransactionActivity="Request Purchase Order"
```

```
6546     tp:requestOrResponseAction="Purchase Order Request
```

```
6547 Action"/>
```

```
6548     <tp:ChannelId>syncChannelA1</tp:ChannelId>
```

```
6549 </tp:ThisPartyActionBinding>
```

```
6550 <tp:CanReceive>
```

Collaboration-Protocol Profile and Agreement Specification

Page

131 of 156

```

6551     <tp:ThisPartyActionBinding
6552         tp:id="companyA_ABID7"
6553         tp:action="Purchase Order Confirmation Action"
6554         tp:packageId="CompanyA_SyncReplyPackage">
6555     <tp:BusinessTransactionCharacteristics
6556         tp:isNonRepudiationRequired="true"
6557         tp:isNonRepudiationReceiptRequired="true"
6558         tp:isConfidential="transient"
6559         tp:isAuthenticated="persistent"
6560         tp:isTamperProof="persistent"
6561         tp:isAuthorizationRequired="true"
6562         tp:timeToAcknowledgeReceipt="PT2H"
6563         tp:timeToPerform="P1D"/>
6564     <tp:ActionContext
6565         tp:binaryCollaboration="Request Purchase Order"
6566         tp:businessTransactionActivity="Request Purchase
6567 Order"
6568         tp:requestOrResponseAction="Purchase Order
6569 Confirmation Action"/>
6570     <tp:ChannelId>syncChannelA1</tp:ChannelId>
6571 </tp:ThisPartyActionBinding>
6572 </tp:CanReceive>
6573 <tp:CanReceive>
6574     <tp:ThisPartyActionBinding
6575         tp:id="companyA_ABID8"
6576         tp:action="Exception"
6577         tp:packageId="CompanyA_ExceptionPackage">
6578     <tp:BusinessTransactionCharacteristics
6579         tp:isNonRepudiationRequired="true"
6580         tp:isNonRepudiationReceiptRequired="true"
6581         tp:isConfidential="transient"
6582         tp:isAuthenticated="persistent"
6583         tp:isTamperProof="persistent"
6584         tp:isAuthorizationRequired="true"
6585         tp:timeToAcknowledgeReceipt="PT2H"
6586         tp:timeToPerform="P1D"/>
6587     <tp:ChannelId>syncChannelA1</tp:ChannelId>
6588 </tp:ThisPartyActionBinding>
6589 </tp:CanReceive>
6590 </tp:CanSend>

```

6591 This subordination will also carry over to the synchronous receiving side, in which any of its
6592 **CanSend** elements are under the **CanReceive** element used to represent the initial receiving of a
6593 request. An illustration from Company B's synchronous binding is:

```

6596 <tp:CanReceive>
6597     <tp:ThisPartyActionBinding
6598         tp:id="companyB_ABID8"
6599         tp:action="Purchase Order Request Action"
6600         tp:packageId="CompanyB_SyncReplyPackage">
6601     <tp:BusinessTransactionCharacteristics

```

```
6602         tp:isNonRepudiationRequired="true"
6603         tp:isNonRepudiationReceiptRequired="true"
6604 tp:isConfidential="transient"
6605         tp:isAuthenticated="persistent"
6606 tp:isTamperProof="persistent"
6607         tp:isAuthorizationRequired="true"
6608 tp:timeToAcknowledgeReceipt="PT5M"
6609         tp:timeToPerform="PT5M"/>
6610 <tp:ActionContext
6611     tp:binaryCollaboration="Request Purchase Order"
6612     tp:businessTransactionActivity="Request Purchase Order"
6613     tp:requestOrResponseAction="Purchase Order Request
6614 Action"/>
6615 <tp:ChannelId>syncChannelB1</tp:ChannelId>
6616 </tp:ThisPartyActionBinding>
6617 <tp:CanSend>
6618     <tp:ThisPartyActionBinding
6619         tp:id="companyB_ABID6"
6620         tp:action="Purchase Order Confirmation Action"
6621         tp:packageId="CompanyB_ResponsePackage">
6622         <tp:BusinessTransactionCharacteristics
6623             tp:isNonRepudiationRequired="true"
6624             tp:isNonRepudiationReceiptRequired="true"
6625             tp:isConfidential="transient"
6626             tp:isAuthenticated="persistent"
6627             tp:isTamperProof="persistent"
6628             tp:isAuthorizationRequired="true"
6629             tp:timeToAcknowledgeReceipt="PT5M"
6630             tp:timeToPerform="PT5M"/>
6631         <tp:ActionContext
6632             tp:binaryCollaboration="Request Purchase Order"
6633             tp:businessTransactionActivity="Request Purchase Order"
6634             tp:requestOrResponseAction="Purchase Order Confirmation
6635 Action"/>
6636         <tp:ChannelId>syncChannelB1</tp:ChannelId>
6637         </tp:ThisPartyActionBinding>
6638     </tp:CanSend>
6639 <tp:CanSend>
6640     <tp:ThisPartyActionBinding
6641         tp:id="companyB_ABID7"
6642         tp:action="Exception"
6643         tp:packageId="CompanyB_ExceptionPackage">
6644     <tp:BusinessTransactionCharacteristics
6645         tp:isNonRepudiationRequired="true"
6646         tp:isNonRepudiationReceiptRequired="true"
6647         tp:isConfidential="transient"
```

```

6648         tp:isAuthenticated="persistent"
6649         tp:isTamperProof="persistent"
6650         tp:isAuthorizationRequired="true"
6651         tp:timeToAcknowledgeReceipt="PT5M"
6652         tp:timeToPerform="PT5M"/>
6653     <tp:ChannelId>syncChannelB1</tp:ChannelId>
6654     </tp:ThisPartyActionBinding>
6655 </tp:CanSend>
6656 </tp:CanReceive>

```

6657

6658 **E.5.2 Matching and Checking DeliveryChannel Details**

6659 Until now, most of the matching work has been undertaken to find pairs of correlative action
6660 binding, and so the matching has functioned as a filtering mechanism. Once in possession of
6661 pairs of correlative action bindings, however, the work of checking for matches across the
6662 various dimensions of operation—transport, transport security, PKI compatibility for various
6663 tasks, agreement about messaging characteristics (reliable messaging, digital enveloping, signed
6664 acknowledgments (minimal non-repudiation of receipt), non-repudiation of origin, packaging
6665 details, and more begins.

6666

6667 Once in possession of the action bindings, IDREFs provide references to the underlying
6668 components for comparison. For example, when comparing packaging details, the *Request*
6669 IDREFS are found at *CanSend/ThisPartyActionBinding/@packageId* and within the other *CPP*
6670 at *CanReceive/ThisPartyActionBinding@packageId*. For Company A's *Request* "Purchase
6671 Order Request Action," the packaging IDREF is found in:

6672

```
6673 tp:packageId="CompanyA_RequestPackage"
```

6674

6675 and this IDREF value refers to:

6676

```

6677 <tp:Packaging tp:id="CompanyA_RequestPackage">
6678     <tp:ProcessingCapabilities tp:parse="true"
6679 tp:generate="true"/>
6680     <tp:CompositeList>
6681         <tp:Composite tp:id="CompanyA_RequestMsg"
6682 tp:mimetype="multipart/related"
6683 tp:mimeparameters="type=text/xml;">
6684         <tp:Constituent tp:idref="CompanyA_MsgHdr"/>
6685         <tp:Constituent tp:idref="CompanyA_Request"/>
6686     </tp:Composite>
6687 </tp:CompositeList>
6688 </tp:Packaging>

```

6689

6690 For Company A's *Request* "Purchase Order Request Action", the delivery channel IDREF is
6691 found in:

6692

```
6693 <tp:ChannelId>asyncChannelA1</tp:ChannelId>
```

6694
6695 and this IDREF value refers to the element with this ID, namely:
6696

```
6697 <tp:DeliveryChannel tp:channelId="asyncChannelA1"
6698 tp:transportId="transportA1" tp:docExchangeId="docExchangeA1">
6699 <tp:MessagingCharacteristics
6700     tp:syncReplyMode="none"
6701     tp:ackRequested="always"
6702     tp:ackSignatureRequested="always"
6703     tp:duplicateElimination="always"/>
6704 </tp:DeliveryChannel>
```

6705
6706 Two remaining crucial references for understanding the binding, are found in attributes of the
6707 *DeliveryChannel*, namely: *DeliveryChannel/@transportId* and in the attribute
6708 *DeliveryChannel/@docExchangeId*.

6709
6710 For Company A, for example, we find *transportId*="transportA1" and
6711 *docExchangeId*="docExchangeA1" are the IDREFs for the continuing binding information with
6712 the *DeliveryChannel*, "asyncChannelA1". Resolving these references, we obtain:

```
6713  
6714 <tp:Transport tp:transportId="transportA1">
6715   <tp:TransportSender>
6716     <tp:TransportProtocol
6717       tp:version="1.1">HTTP</tp:TransportProtocol>
6718     <tp:TransportClientSecurity>
6719       <tp:TransportSecurityProtocol
6720         tp:version="3.0">SSL</tp:TransportSecurityProtocol>
6721       <ClientCertificateRef tp:certId="CompanyA_ClientCert"/>
6722       <tp:ServerSecurityDetailsRef
6723         tp:securityId="CompanyA_TransportSecurity"/>
6724     </tp:TransportClientSecurity>
6725   </tp:TransportSender>
6726   <tp:TransportReceiver>
6727     <tp:TransportProtocol
6728       tp:version="1.1">HTTP</tp:TransportProtocol>
6729     <tp:Endpoint
6730       tp:uri="https://www.CompanyA.com/servlets/ebxmlhandler/asyn
6731       c" tp:type="allPurpose"/>
6732     <tp:TransportServerSecurity>
6733       <tp:TransportSecurityProtocol
6734         tp:version="3.0">SSL</tp:TransportSecurityProtocol>
6735       <tp:ServerCertificateRef tp:certId="CompanyA_ServerCert"/>
6736       <tp:ClientSecurityDetailsRef
6737         tp:securityId="CompanyA_TransportSecurity"/>
6738     </tp:TransportServerSecurity>
6739   </tp:TransportReceiver>
```

```

6740 </tp:Transport>
6741
6742 for transportID "transportA1" and
6743
6744 <tp:DocExchange tp:docExchangeId="docExchangeA1">
6745   <tp:ebXMLSenderBinding tp:version="2.0">
6746     <tp:ReliableMessaging>
6747       <tp:Retries>3</tp:Retries>
6748       <tp:RetryInterval>PT2H</tp:RetryInterval>
6749       <tp:MessageOrderSemantics>Guaranteed</tp:MessageOrderSemant
6750 ics>
6751     </tp:ReliableMessaging>
6752     <tp:PersistDuration>P1D</tp:PersistDuration>
6753     <tp:SenderNonRepudiation>
6754       <tp:NonRepudiationProtocol>http://www.w3.org/2000/09/xmldsi
6755 g#
6756       </tp:NonRepudiationProtocol>
6757       <tp:HashFunction>http://www.w3.org/2000/09/xmlldsig#sha1
6758     </tp:HashFunction>
6759     <tp:SignatureAlgorithm>http://www.w3.org/2000/09/xmlldsig#ds
6760 a-sha1
6761     </tp:SignatureAlgorithm>
6762     <tp:SigningCertificateRef
6763 tp:certId="CompanyA_SigningCert"/>
6764     </tp:SenderNonRepudiation>
6765     <tp:SenderDigitalEnvelope>
6766     <tp:DigitalEnvelopeProtocol
6767 tp:version="2.0">S/MIME</tp:DigitalEnvelopeProtocol>
6768     <tp:EncryptionAlgorithm>DES-CBC</tp:EncryptionAlgorithm>
6769     <tp:EncryptionSecurityDetailsRef
6770 tp:securityId="CompanyA_MessageSecurity"/>
6771     </tp:SenderDigitalEnvelope>
6772     </tp:ebXMLSenderBinding>
6773     <tp:ebXMLReceiverBinding tp:version="2.0">
6774     <tp:ReliableMessaging>
6775       <tp:Retries>3</tp:Retries>
6776       <tp:RetryInterval>PT2H</tp:RetryInterval>
6777       <tp:MessageOrderSemantics>Guaranteed</tp:MessageOrderSemant
6778 ics>
6779     </tp:ReliableMessaging>
6780     <tp:PersistDuration>P1D</tp:PersistDuration>
6781     <tp:ReceiverNonRepudiation>
6782       <tp:NonRepudiationProtocol>http://www.w3.org/2000/09/xmldsi
6783 g#
6784       </tp:NonRepudiationProtocol>
6785       <tp:HashFunction>http://www.w3.org/2000/09/xmlldsig#sha1

```



```

6786     </tp:HashFunction>
6787     <tp:SignatureAlgorithm>http://www.w3.org/2000/09/xmlsig#ds
6788 a-sha1
6789     </tp:SignatureAlgorithm>
6790     <tp:SigningSecurityDetailsRef
6791 tp:securityId="CompanyA_MessageSecurity"/>
6792 </tp:ReceiverNonRepudiation>
6793 <tp:ReceiverDigitalEnvelope>
6794 <tp:DigitalEnvelopeProtocol
6795 tp:version="2.0">S/MIME</tp:DigitalEnvelopeProtocol>
6796 <tp:EncryptionAlgorithm>DES-CBC</tp:EncryptionAlgorithm>
6797 <tp:EncryptionCertificateRef
6798 tp:certId="CompanyA_EncryptionCert"/>
6799 </tp:ReceiverDigitalEnvelope>
6800 </tp:ebXMLReceiverBinding>
6801 </tp:DocExchange>

```

6802
6803 for the *docExchangeId*, docExchangeA1.

6804
6805 There are, of course, other references, such as those to security-related capabilities, that will be
6806 important to resolve when checking detailed matching properties, but the four IDREFs (two for
6807 the sender and two for the receiver) that have just been introduced are critical to the remainder of
6808 the match tests that will lead to the formation of draft *CPAs*. We will assume at this point that the
6809 reader can resolve IDREFs using the example *CPPs* and *CPAs* for Company A and B in the
6810 appendices, and will not exhibit them in the text in order to save space.

6811
6812 We next turn to a more in-depth treatment of the tests that are involved in finding the elements
6813 for a draft *CPA*.

6814
6815 The detailed tasks to be discussed in greater depth are:

- 6816
- 6817 1. Matching *Channel Messaging Characteristics*
 - 6818 2. Checking *Business Transaction Characteristics* coherence with *Channel* details
 - 6819 3. Matching *Packaging*
 - 6820 4. Matching *Transport* and *Transport[Receiver|Sender]Security*
 - 6821 5. Matching and Checking *DocExchange* subtrees.

6822
6823 Because agreement about *Transport* is quite fundamental, we shall consider it first.
6824 Computational processes are likely to first find pairs that match on *Transport* details, and will
6825 ignore pairs failing to have matches at this level.

6826 6827 **E.5.2.1 Matching Transport**

6828 Matching *Transport* first involves matching the *Transport/TransportSender/TransportProtocol*
6829 capabilities of the requester with the *Transport/TransportReceiver/TransportProtocol*
6830 capabilities found under the collaborator receiving the *request*. Several such matches can exist,

6831 and any of these matches can be used in forming a draft, provided other aspects match up
6832 satisfactorily. Each *CPP* is assumed to have listed its preferred transport protocols first (as
6833 determined by the listing of the Bindings that reference the **Transport** element, but different
6834 outcomes can result depending on which *CPP* is used first for searching for matches. In general,
6835 resolution of preference differences is left to a distinct phase of *CPA negotiation*, following
6836 proposal of a draft *CPA*. Negotiation can be performed by explicit actions of users, but is
6837 expected to become increasingly automated.

6838
6839 Matching transport secondly involves matching the **TransportSender/TransportProtocol**
6840 capabilities of the responding collaborator with its **TransportReceiver/TransportProtocol**
6841 capabilities found under the collaborator receiving the response, which is typically the
6842 collaborator that has sent a request. Several such matches can exist, and any of these matches can
6843 be used in forming a draft. In one case, however, there may be no need for the second match on
6844 **TransportProtocol**. If we are using HTTP or some other protocol supporting synchronous
6845 replies, and the **DeliveryChannel** has a **MessagingCharacteristics** child that has its
6846 **syncReplyMode** attribute with a value of “signalsAndResponse,” then everything comes back
6847 synchronously, and there is no need to match on **TransportProtocol** for the **Response**
6848 **DeliveryChannel**.

6849
6850 If **TransportSecurity** is present, then there can be additional checks. First,
6851 **TransportSender/TransportClientSecurity/TransportSecurityProtocol** should be compatible
6852 with **TransportReceiver/TransportServerSecurity/TransportSecurityProtocol**. Second, if either
6853 the **TransportSender/TransportClientSecurity/ClientSecurityDetailsRef** or
6854 **TransportSender/TransportClientSecurity/ServerSecurityDetailsRef** elements are present, and
6855 the IDREF references an element containing some **AnchorCertificateRef**, then an opportunity
6856 exists to check suitability of one *Party*'s PKI trust of the certificates used in the
6857 **TransportSecurityProtocol**. For example, by resolving the IDREF value in
6858 **TransportSender/TransportClientSecurity/ClientCertificateRef/@certId**, we can obtain the
6859 proposed client certificate to use for client-side authentication. By resolving the IDREFs from
6860 the **AnchorCertificateRef**, we become able to determine whether the proposed client certificate
6861 will “chain to a trusted root” on the server side's PKI. Similar remarks apply to checks on the
6862 validity of a server certificate found by resolving
6863 **TransportReceiver/TransportServerSecurity/ServerCertificateRef**. This server certificate can
6864 be checked against the CA trust anchors that are found by resolving
6865 **TransportSender/TransportClientSecurity/ServerSecurityDetailsRef/@securityId**, and finding
6866 CA certificates (or CA certificate chains) in the **KeyInfo** elements under the **Certificate** element
6867 obtained by resolving the IDREF found in **AnchorCertificateRef@certId**.

6868
6869 When matches exist for the correlative **Transport** components, we then have discovered an
6870 interoperable solution at the transport level. If not, no *CPA* will be available, and a gap has been
6871 identified that will need to be remedied by whatever exception handling procedures are in place.
6872 Let us next consider other capabilities that need to match for “thicker” interoperable solutions.

6873
6874 **E.5.2.2 Checking BusinessTransactionCharacteristics and DeliveryChannel**
6875 **MessagingCharacteristics**

6876 Under each of the correlative action bindings, there is a child element of ***DeliveryChannel***,
6877 ***MessagingCharacteristics*** that has several attributes important in *CPA* formation tasks. The
6878 attributes having wider implications are ***syncReplyMode***, ***ackRequested***, and
6879 ***ackSignatureRequested***; for the ***duplicateElimination*** and ***actor*** attributes, compatibility exists
6880 when the attributes that are found under the ***CanSend*** and ***CanReceive DeliveryChannels*** have
6881 the same values. As the element's name implies, all of these ***DeliveryChannel*** features pertain to
6882 the messaging layer.

6883
6884 In addition, ***BusinessTransactionCharacteristics***, found under ***ThisPartyActionBinding***,
6885 contains attributes reflecting a variety of features pertaining to desired security and business
6886 transaction properties that are to be implemented by the agreed upon ***DeliveryChannels***. These
6887 properties may have implications on what capabilities are needed within more detailed
6888 components of the ***DeliveryChannel*** elements, such as in the ***Packaging*** element. When using a
6889 *BPSS* process specification, these properties may be specified within the BusinessTransaction.
6890 The properties of the ***BusinessTransactionCharacteristics*** element are, however, the ones that
6891 will be operative in the implementation of the ***BusinessTransaction***, and may override the
6892 specified values found in the *BPSS* Process specification. Because the properties are diverse, the
6893 details that implement the properties can be spread over other elements referenced within the
6894 ***DeliveryChannel*** elements.

6895
6896 These attributes apply to either a *Request* or *Response* delivery channel, but can impact either the
6897 *Sender* or *Receiver* (or both) in a channel. In addition, the attributes governing
6898 acknowledgments, for example, qualify the interrelation of ***DeliveryChannel*** elements by
6899 specifying behavior that is to occur that qualifies the contents of a return message.

6900
6901 The most basic test for compatibility for any of the attributes in either ***MessagingCharacteristics***
6902 or ***BusinessTransactionCharacteristics*** is that the attributes are equal in the sending party's
6903 ***DeliveryChannel*** referenced by ***CanSend/ThisPartyActionBinding/ChannelId*** and in the
6904 receiving party's ***DeliveryChannel*** referenced by
6905 ***CanReceive/ThisPartyActionBinding/ChannelId***. If they are unequal, and all Bindings have
6906 been examined on both sides, a draft *CPA* will represent a compromise to some common set with
6907 respect to the functionality represented by the attributes.

6908
6909 In the following discussions, we will consider many of the attributes in the two *Characteristics*
6910 elements, and relate them to additional underlying implementational details, one of which is
6911 ***Packaging***.

6912
6913 From a high level, basic agreement in *packaging* is a matter of compatibility of the generated
6914 *packaging* on the sending side with the parsed *packaging* on the receiving side. The basic
6915 packaging check is, therefore, checking packaging compatibility under the ***CanSend*** element of a
6916 sender action with the packaging under the ***CanReceive*** element of that same action under the
6917 receiver side.
6918

6919 For efficiency, representation of capabilities of parsing/handling packaging can make use of both
 6920 wildcards and repetition, and as needed these capabilities can also express open data formatting
 6921 used on the generating side. For example, consider the **SimplePart**:

```
6922 <tp:SimplePart tp:id="IWild" tp:mimetype="*/"/>
```

6925 By wildcarding **mimetype** values, we represent our capability of accepting any data, and would
 6926 match any specific MIME type. Also, consider a **Constituent** appearing within a **Composite**:

```
6927 <tp:Constituent tp:idref="MsgHdr"/>
6928 <tp:Constituent minOccurs="0"
6929 maxOccurs="10" tp:idref="IWild"/>
```

6931 This notation serves to capture the capability of handling any number of arbitrary MIME
 6932 bodyparts within the **Composite** being defined. A Packaging capability such as this would
 6933 obviously match numerous more specific generated *Packaging* schemes, as well as matching
 6934 literally with a scheme of the same generality.

6937 Certain more complex checks are needed for more complicated packaging options pertaining to
 6938 syncReplyMode. These are discussed in the following.

6939 **syncReplyMode**

6941 The **syncReplyMode** has a value other than “none” to indicate what parts of a message should be
 6942 returned in the *Reply* of a transport capable of synchronous operation, such as HTTP. (We here
 6943 use “synchronous” to mean “on the same TCP connection,” which is one use of this term. We do
 6944 not specify any waiting, notification, or blocking behavior on processes or threads that are
 6945 involved, though presumably there is some computational activity that maintains the connection
 6946 state and is above the TCP and socket layers.)

6947 The possible implementations pertaining to various values of the **syncReplyModes** are numerous,
 6948 but we will try to indicate at least the main factors that are involved.

6950 As will be seen, the **Packaging** element is important in specifying implementation details and
 6951 compatibilities. But, because business level signals may be involved, other action bindings may
 6952 need examination in addition to the already selected bindings for the *Request* and *Response*.
 6953 Also, the values of **TransportReceiver/Endpoint/@type** might need checking when producing
 6954 draft *CPAs*.

- 6957 • Let us first begin with the cases in which *Responses*, *Message Service Handler Signals*
 6958 and *Business Signals* return in some combination of a synchronous reply and other
 6959 asynchronous message(s). These various combinations will be discussed for the
 6960 **syncReplyMode** values: "mshSignalsOnly," "signalsOnly," "responseOnly," and
 6961 "signalsAndResponse."
- 6962 •
- 6963 • By convention, synchronous replies are represented by subordinating **CanSend** or

6964 *CanReceive* elements under the *CanReceive* or *CanSend* elements that represent the initial
6965 *Request* binding capabilities. For representing asynchronous Requests, Replies, or Signals,
6966 the *CanSend* or *CanReceive* elements are all siblings and directly subordinate to the
6967 *ServiceBinding*. Therefore, both asynchronous and synchronous capabilities can be grouped
6968 under a *ServiceBinding* in a *CPP*, and can still be unambiguously distinguished. In principle,
6969 increasing subordination (nesting) can indicate patterns of dialog more elaborate than
6970 *Request* and *Response*. Few use cases for this functionality are common at the time of this
6971 writing.

6972
6973 ***mshSignalsOnly***

6974 The Request sender's *DeliveryChannel* (referenced by
6975 *CanSend/ThisPartyActionBinding/ChannelId*) and the Request receiver's *DeliveryChannel*
6976 (referenced by *CanReceive/ThisPartyActionBinding/ChannelId*) both should have
6977 *MessagingCharacteristics/@syncReplyMode* value of *mshSignalsOnly*.

6978
6979 While a Party can explicitly identify a *DeliveryChannel* for the SOAP envelope with subordinate
6980 *CanSend* and *CanReceive* elements, and with them specialized bindings, these are typically
6981 omitted for ebXML Messaging software. It is presumed that each side can process a synchronous
6982 reply constructed in accordance with ebXML Messaging. The *DeliveryChannel* representation
6983 mechanism here serves as a placeholder for capturing other Messaging Signal protocols that
6984 might emerge.

6985
6986 Currently acknowledgments and signed acknowledgments, along with errors, are the primary
6987 MSH signals that are included in the SOAP envelope. If Company A set
6988 *syncReplyMode* to *mshSignalsOnly*, then Company B's correlative
6989 *CanReceive/ThisPartyActionBinding/@packageId* should contain a nested
6990 *CanSend/ThisPartyActionBinding/@packageId* for a message without any business payload or
6991 signals. In addition, the *CanSend/ThisPartyActionBinding/@packageId* of Company B's
6992 *Response* should resolve to packaging format capable of returning the *Response* (and possibly
6993 other constituents) asynchronously. The compatibility of the *DeliveryChannel* elements can be
6994 checked, as can the capability of Company A to receive that Response payload, the Signal
6995 payload(s), or Responses bundled with signals as specified by the packaging formats that are
6996 referenced through the relevant *ThisPartyActionBindings* element's *packageId* attribute values.

6997
6998 ***signalsOnly***

6999 The Request sender's *DeliveryChannel* (referenced by its
7000 *CanSend/ThisPartyActionBinding/ChannelId*) and the Request receiver's *DeliveryChannel*
7001 (referenced by its *CanReceive/ThisPartyActionBinding/ChannelId*) both should have
7002 *MessagingCharacteristics/@syncReplyMode* value of *signalsOnly*.

7003
7004 If Company A sets *syncReplyMode* to *signalsOnly*, then under Company B's correlative
7005 *CanReceive* element, there should be a nested *CanSend/ThisPartyActionBinding* whose
7006 *packageId* attribute's value resolves to a packaging format appropriate for Signals. For the
7007 *CanSend/ThisPartyActionBinding/@packageId* associated with Company B's business level
7008 *Response*, the attribute IDREF value should resolve to a packaging format capable of returning

7009 payloads and that omits business signals. This **CanSend** element will be a direct child of
7010 **ServiceBinding**, a placement representing its asynchronous character. The original requesting
7011 party will need to have a **CanReceive/ThisPartyActionBinding** that is compatible with the
7012 responding party, and that is a direct child of its **ServiceBinding** element.

7013
7014 Using subordinate **CanSend** and subordinate **CanReceive** elements can be useful if the
7015 **DeliveryChannel** details for Exception signals differ from those specified for Request and
7016 Response. Signal bindings, for example, may differ by omitting **ackRequested**, or possibly one
7017 of the security features (digital enveloping or non-repudiation of receipt) that are used for
7018 Requests or Responses. Just as with other tests on Requests and Responses, there can be checks
7019 for compatibility in **Packaging**, **DocExchange**, **MessagingCharacteristics**, or
7020 **BusinessTransactionCharacteristics** referred to in the correlative subordinate **CanSend** and
7021 **CanReceive DeliveryChannels**.

7022
7023 **responseOnly**

7024 The Request sender's **DeliveryChannel** (referenced by
7025 **CanSend/ThisPartyActionBinding/ChannelId**) and the Request receiver's **DeliveryChannel**
7026 (referenced by **CanReceive/ThisPartyActionBinding/ChannelId**) both should have
7027 **MessagingCharacteristics/@syncReplyMode** value of **responseOnly**.

7028
7029 If Company A sets **syncReplyMode** to **responseOnly**, the
7030 **CanSend/ThisPartyActionBinding/@packageId** of Company B's response should resolve to a
7031 packaging format capable of returning payloads, but omitting business signals. The
7032 **CanSend/ThisPartyActionBinding** element will be included as a child of the **CanReceive**
7033 element so the responder can indicate that it is a synchronous response.

7034
7035 There should be an independent way to return business level error signals. So, there should be a
7036 **ThisPartyActionBinding** for any Signal payload announced, and these bindings should be at the
7037 direct child of **ServiceBinding** level to represent their asynchronous flavor.

7038
7039 It is not too likely that **ReceiptAcknowledgment** and similar signals will be used when a response
7040 is returned synchronously. The motivation for using these signals is indicating positive forward
7041 progress, and this motivation will be undermined when a Response is returned directly.

7042
7043 For the **responseOnly** case, including subordinate **CanSend/ThisPartyActionBinding** and
7044 **CanReceive/ThisPartyActionBinding**, means that there can be checks for compatibility in
7045 **Packaging**, **DocExchange**, **MessagingCharacteristics**, or **BusinessTransactionCharacteristics**.
7046 The **syncReplyMode** and **ackRequested** attributes here should be carefully considered because a
7047 **mshSignalsOnly** value here would mean that another round of synchronous messaging will need
7048 to occur on the same connection. Incidentally, for **Transport** elements referenced under
7049 subordinate bindings, there need not be any **Endpoint** elements. If there are **Endpoint** elements,
7050 they may be ignored.

7051
7052 **signalsAndResponse**

7053 The Request sender's *DeliveryChannel* (referenced by
7054 *CanSend/ThisPartyActionBinding/ChannelId*) and the Request receiver's *DeliveryChannel*
7055 (referenced by *CanReceive/ThisPartyActionBinding/ChannelId*) both should have
7056 *MessagingCharacteristics/@syncReplyMode* value of *signalsAndResponse*.

7057
7058 If Company A sets *syncReplyMode* to *signalsAndResponse*, the
7059 *CanSend/ThisPartyActionBinding* of Company B's response should be subordinate to Company
7060 B's *CanReceive* element. The packaging format that is referenced should be capable of
7061 returning payloads and signals bundled together. If no asynchronous bindings exist for error
7062 signals, this will be the only defined *DeliveryChannel* agreed to for all aspects of message
7063 exchange for the business transaction. However, it is likely that an asynchronous binding would
7064 normally be provided to send Exception signals.

7065
7066 ***ackRequested and ackSignatureRequested***
7067 Checks on the *ackRequested* and *ackSignatureRequested* attributes within correlative
7068 *DeliveryChannels* (that is, correlative because referenced under one action's *CanSend* and
7069 *CanReceive* elements) are primarily to see that the values of the corresponding attributes are the
7070 same.

7071
7072 However, there are some interactions of these attributes with other information items that need to
7073 be mentioned.

7074
7075 The principal use of the *ackRequested* attribute is within reliable messaging configurations. If
7076 reliable messaging is to be configured, then checks on agreement in the correlative
7077 *ReliableMessaging* elements as found under *DocExchange/ebXMLSenderBinding* and
7078 *DocExchange/ebXMLReceiverBinding* are in order. Also, the value of the
7079 *duplicateElimination* attribute of *MessagingCharacteristics* should be checked for agreement.
7080 Draft *CPAs* may be formed by deliberately aligning values that are not equal along some of these
7081 dimensions. Downgrading may provide draft *CPAs* most likely to gain acceptance; so, for
7082 example, if *duplicateElimination* is false on the receiving side, aligning it to false on the sending
7083 side is most likely to produce a draft that succeeds.

7084
7085 The additional function of *ackSignatureRequested* is that it provides a "thin" implementation for
7086 *non-repudiation of receipt*. The basic check is for equality of attribute value, but additional
7087 constraints may need test and alignment. If no signal capable of implementing *non-repudiation*
7088 *of receipt* is found under the *ServiceBinding*, then having an "always" value for
7089 *ackSignatureRequested* suggests aligning the *BusinessTransactionCharacteristics* attributes,
7090 *isNonRepudiationReceiptRequired*, to be true. However, if this is done, care should be taken to
7091 check that the *BusinessTransactionCharacteristics* attribute *isIntelligibleCheckRequired* is
7092 false. This is because the messaging implementation only deals with receipt in the sense of
7093 having received a byte stream off the wire (and persisting it so that it is available for further
7094 processing). It is not safe to presume that any syntactical or semantic checks on the data were
7095 performed.

7096
7097 **E.5.2.3 DocExchange Checks for BusinessTransactionCharacteristics**

7098 When using *CPPs* and *CPAs* with ebXML Messaging, which is the most likely early deployment
7099 situation, there exists an opportunity to check agreement on ***BusinessTransactionCharacteristics***
7100 attributes:

7101
7102 The following three attributes need to have equal values in the bindings for a Request or for a
7103 Response. No further discussion will be provided in this appendix on these “deadlines,” except to
7104 say that a sophisticated proposed *CPA* generation tool might check on the coherence of the
7105 values chosen here with values for reliable messaging parameters, existence of compatible
7106 ReceiptAcknowledgment or AcceptanceAcknowledgment bindings, and consistency with
7107 syncReplyMode internal configuration.

```
7108  
7109 <attribute name="timeToAcknowledgeReceipt" type="duration" />  
7110 <attribute name="timeToAcknowledgeAcceptance" type="duration" />  
7111 <attribute name="timeToPerform" type="duration" />
```

7112
7113 The remaining attributes involve a number of security related issues and will be the focus of the
7114 remaining discussion of ***BusinessTransactionCharacteristics*** attributes:

```
7115  
7116 <attribute name="isNonRepudiationRequired" type="boolean" />  
7117 <attribute name="isNonRepudiationReceiptRequired"  
7118 type="boolean" />  
7119 <attribute name="isIntelligibleCheckRequired" type="boolean" />  
7120 <attribute name="isAuthenticated"  
7121 type="tns:persistenceLevel.type" />  
7122 <attribute name="isTamperProof"  
7123 type="tns:persistenceLevel.type" />  
7124 <attribute name="isAuthorizationRequired" type="boolean" />  
7125 <attribute name="isConfidential"  
7126 type="tns:persistenceLevel.type" />
```

7127 Here, the basic test is that for correlative ***DeliveryChannels***, the corresponding attributes have
7128 the same values. Again there are some interaction aspects with parts of the ***DeliveryChannel*** that
7129 motivate making some additional checks.

7130
7131 Previously, when discussing the ***MessagingCharacteristics*** attribute ***ackSignatureRequested***, it
7132 was pointed out that the messaging implementation provides thin support for holding
7133 ***isNonRepudiationReceiptRequired*** true provided that the attribute ***isIntelligibleCheckRequired***
7134 is false. When both are true, then there should exist a business signal with compatible ***Packaging***
7135 and ***DeliveryChannel*** values. If the signal has been independently described within
7136 asynchronous ***CanSend*** and ***CanReceive*** elements, knowing the signal name (such as,
7137 “ReceiptAcknowledgment”) may support a relatively simple search and test. However, if
7138 synchronous transports are involved, some filters using ***syncReplyModes*** may be needed to
7139 discover an underlying support for a “thick” implementation of *non-repudiation of receipt*.

7140
7141 When non-repudiation of receipt is implemented by a business signal, then checks on signing
7142 certificate validity can involve the ***CollaborationRole/ApplicationCertificateRef*** and the
7143 ***CollaborationRole/ApplicationSecurityDetailsRef***, that provides a reference to the
7144 ***SecurityDetails*** element containing the list of ***TrustAnchors***. The certificate from the side

7145 signing the ReceiptAcknowledgment would be checked against the certificates referred to by the
7146 *AnchorCertificateRef* under *TrustAnchors*.

7147
7148 The business signal will sometimes be conveyed as part of a message. It remains true that the
7149 message itself will still be sent through a MSH, and that the MSH can also sign the message
7150 using the certificate found by resolving the IDREF found at
7151 *DocExchange/ebXMLSenderBinding/SenderNonRepudiation/SigningCertificateRef/@certId*.

7152
7153 If a particular software component implements both MSH functionality and business level
7154 security functionality, it is possible that the same certificate may be pointed to by
7155 *ApplicationCertificateRef* and *SigningCertificateRef/@certId*. In other words, the distinction
7156 between MSH level signing and application level signing is a logical one, and may not
7157 correspond with software component boundaries. Because the MSH signature is over the
7158 message, the message signature may be over an application level signature. While this may be
7159 redundant for some system configurations, protocols may require both signatures to exist over
7160 the different regions.

7161
7162 Failure to validate a certificate may not prevent formation of a draft *CPA*. First, the sender's signing
7163 certificate can be a self-signed certificate. If so, a reference to this self-signed certificate may be
7164 added to the receiver's *TrustAnchors/AnchorCertificateRef* list. This proposal amounts to
7165 proposing to agree to a direct trust model, rather than a hierarchical model involving certificate
7166 authorities. Second, a proposal to add a trusted root may be made, again by appropriate revision of
7167 the *TrustAnchors*.

7168
7169 When non-repudiation of receipt is implemented by the Messaging layer, the checks on PKI
7170 make use of elements under *DocExchange*.

7171
7172 *isNonRepudiationRequired*
7173 *isAuthenticated*
7174 *isAuthorizationRequired*
7175 *isTamperProof*

7176
7177 The ideas of authentication, authorization, nonrepudiation and being "tamper proof" may be very
7178 distinct as business level concepts, yet the implementation of these factors tend to use very
7179 similar technologies. Actually, prevention of tampering is not literally implemented. Instead,
7180 means are provided for detecting that tampering (or some accidental garbling) has occurred.
7181 Likewise, implementations of authorization usually are provided by implementations of access
7182 control (permitting or prohibiting a user in a role making use of a resource) and presentation of a
7183 token or credential to gain access, which may involve authentication as an initial step!
7184 Nonrepudiation may build on all the previous functions, plus retaining information for supplying
7185 presumptive evidence of origination at some later time.

7186
7187 When checking whether *isNonRepudiationRequired* can be set to True for both *Parties*, check
7188 whether the signing certificate will be counted as valid at the receiver.

7189 The IDREF reference to the signing certificate is found in
7190 ***DocExchange/ebXMLSenderBinding/SenderNonRepudiation/SigningCertificateRef/@certId.***
7191 The referenced certificate should be checked for validity with respect to the trust anchors
7192 obtained from ***TrustAnchors/AnchorCertificateRef*** elements under the ***SecurityDetails***
7193 element referenced by the IDREF at
7194 ***DocExchange/ebXMLReceiverBinding/ReceiverNonRepudiation/SigningSecurityDetailsRef/***
7195 ***@securityId.***

7196
7197 As previously noted, failure to validate a certificate does not prevent constructing a draft CPA.
7198 Either self-signed certificates or new trust anchors can be added to align the trust model on one
7199 side with the other side's certificate.

7200
7201 In addition to checking the interoperability of the PKI infrastructures, checks on compatibility of
7202 values in the other attributes in
7203 ***DocExchange/ebXMLReceiverBinding/ReceiverNonRepudiation*** and in
7204 ***DocExchange/ebXMLSenderBinding/SenderNonRepudiation*** can be made.
7205 ***NonRepudiationProtocol***, ***HashFunction***, and ***SignatureAlgorithm*** values may be compatible
7206 even when not equal if knowledge of the protocol requirements allows fallback to a mandatory to
7207 implement value. So values here can be found equal, aligned, or negotiated to reach an
7208 agreement.

7209
7210 If ***isNonRepudiationRequired*** is True, the ***isAuthenticated*** and ***isTamperProof*** should also be
7211 True. This is because in implementing ***isNonRepudiationRequired*** by means of a digital
7212 signature, both authentication (with respect to the identity associated with the signing certificate)
7213 and tamper detection (with respect to the cryptographic hash of the signature) will be
7214 implemented as well. The converses need not be true because authentication and tamper
7215 detection might be accomplished without archiving information needed to support claims of
7216 nonrepudiation.

7217
7218 ***isConfidential***

7219
7220
7221 The ***isConfidential*** attribute indicates properties variously distributed among levels of the
7222 application-to-application sending/receiving stacks.

7223
7224 ***isConfidential*** has possible values of "none", "transient", "persistent", and "transient-and-
7225 persistent. The "persistent" or "transient-and-persistent" values indicate that some digital
7226 enveloping function is present; a "transient" value indicates confidentiality is applied at the
7227 transfer layer or below.

7228
7229 ebXML Messaging version 2.0 does not have an "official" implementation for digital envelopes,
7230 and refers to the future XML Encryption specification as its intended direction for that function.
7231 However, the XML Encryption specification is now a candidate recommendation, and is suitable
7232 for preliminary implementation.

7233

7234 Within the *CPA*, the *DocExchange/ebXMLSenderBinding/SenderDigitalEnvelope* and
7235 *DocExchange/ebXMLReceiverBinding/ReceiverDigitalEnvelope* can provide configuration
7236 details pertaining to security in accordance with [XMLENC]. Use of XML Encryption also will
7237 normally show up in the value of *DigitalEnvelopeProtocol*, and can also appear within a
7238 *NamespaceSupported* element within *Packaging*.

7239
7240 Currently, [ebMS] has only indicated a direction to eventually use XML Encryption, but has not
7241 mandated any digital envelope protocol. Digital enveloping may be done at the “application
7242 level,” and will show up under MIME types within the *Packaging* element. PKI matching will
7243 make use of certificates supplied in *ApplicationCertificateRef* and
7244 *ApplicationSecurityDetailsRef*. If other protocols are to be used, it would be safest to use
7245 extensions to the content model of *DocExchange*, such as, *XXXSenderBinding* and
7246 *XXXReceiverBinding*, and follow the pattern of the ebXML content models for *DocExchange*.
7247 Future versions of this specification intend to make these extension semantics easier to use
7248 interoperably; currently, the extensions would be a multilateral extension within some trading
7249 community.

7250
7251 When checking whether *isConfidential* can be set to “persistent” or “transient-and-persistent”
7252 for both *Parties*, check whether the key exchange certificate will be counted as valid at the
7253 sender. The IDREF reference to the *SecurityDetails* element is found in
7254 *DocExchange/ebXMLSenderBinding/SenderDigitalEnvelope/EncryptionSecurityDetailsRef/@*
7255 *securityId*. The trust anchor certificates obtained from *TrustAnchors/AnchorCertificateRef*
7256 elements under the *SecurityDetails* element will be used to test that the certificate referenced by
7257 *DocExchange/ebXMLReceiverBinding/ReceiverDigitalEnvelope/EncryptionCertificateRef/@c*
7258 *ertId* validates at the sender side.

7259
7260 As previously noted, failure to validate a certificate does not prevent constructing a draft CPA.
7261 Either self-signed certificates or new trust anchors can be added to align the trust model on one
7262 side with the other side’s certificate.

7263
7264 In addition to the PKI related checks and alignments, the elements *EncryptionAlgorithm* and
7265 *DigitalEnvelopeProtocol* should be checked for equality (or compatibility) and, if not
7266 compatible or equal, aligned to values that would work for an initial version of a proposed *CPA*.
7267 Preferences and alignment of these elements can be achieved in a subsequent Negotiation phase.

7268
7269 Finally, it is possible that one side’s DigitalEnvelope will be modeled using either the
7270 *DocExchange/ebXMLSenderBinding/SenderDigitalEnvelope* and
7271 *DocExchange/ebXMLReceiverBinding/ReceiverDigitalEnvelope*, while the other side uses only
7272 *Packaging* to indicate use of, for example, S/MIME Digital Envelopes, because it receives an
7273 already enveloped payload from an application. In such a case, the PKI certificate validation
7274 check could require checking that a certificate described by
7275 *DocExchange/ebXMLReceiverBinding/ReceiverDigitalEnvelope/EncryptionCertificateRef/@c*
7276 *ertId* validates against the *TrustAnchors* found by resolving
7277 *CollaborationRole/ApplicationSecurityDetailsRef*. This complication arises from the possibility

7278 that digital enveloping functionality can be spread over quite distinct portions of the stack in
7279 different software installations.
7280

7281 **E.6 CPA Formation: Technical Details**

7282 When assembling a draft *CPA* from matching portions of two *CPPs*' *PartyInfo* elements, some
7283 additional constraints need to be observed.

7284

7285 First, as mentioned in section 9.11.1, software for producing draft CPAs needs to guarantee that
7286 ID values in one *CPP* are distinct from ID values in the other *CPP* so that no IDREF references
7287 collide when the *CPPs* are merged. The following ID values are potentially subject to collision:

7288

7289 *Certificates*

7290 *SecurityDetails*

7291 *SimplePart*

7292 *Packaging*

7293 *DocExchange*

7294 *Transport*

7295 *DeliveryChannel*

7296 *ThisPartyActionBinding*

7297

7298 There are elements and complex type definitions containing IDREFs. Also some elements have
7299 attributes with IDREF values. These are:

7300

7301 *PartyInfo*

7302 *ActionBinding.type*

7303 *ThisPartyActionBinding*

7304 *OtherPartyActionBinding*

7305 *OverrideMSHActionBinding*

7306 *ChannelId*

7307 *DeliveryChannel*

7308 *Constituent*

7309 *CertificateRef.type*

7310 *AnchorCertificateRef*

7311 *ApplicationCertificateRef*

7312 *ClientCertificateRef*

7313 *ServerCertificateRef*

7314 *SigningCertificateRef*

7315 *EncryptionCertificateRef*

7316 *CertificateRef*

7317 *SecurityDetailsRef.type*

7318

7319 Second, when the *CanSend* and *CanReceive* binding information has been found to match
7320 (equal, correspond with, or be compatible with) the binding information under the other Party's
7321 *CanReceive* and *CanSend* elements, the IDREF references for the *OtherPartyActionBinding*

7322 are filled out in the CPA.

7323

7324 Third, for CPAs that are signed, the implementer is advised to review section 9.9.1.1 when using
7325 [XMLDSIG] for the signature technique. A proposed CPA need not have a signature.

7326

7327 Fourth, when a *CPA* is composed from two *CPPs*, see section 8.8 in which it stated that all
7328 **Comment** elements from both *CPPs* SHALL be included in the *CPA* unless agreed to otherwise.

7329

7330 Fifth, several tests on CPA validity could be conducted on draft CPAs, but these tests are more
7331 critical for a negotiated CPA that is to be deployed and imported into run-time software
7332 components.

7333

7334 1. Expiration: Certificates used in signing a CPA can be checked to verify that they do not expire
7335 before the CPA expires, as given in the **End** element.

7336

7337 2. Certificate expiration: If a CPA lifetime exceeds the lifetime of certificates accepted for use in
7338 signing, key exchange or other security functions, then it would be advisable to make ds:KeyInfo
7339 refer to certificates, rather than to include them within the element by value.

7340

7341 3. Process-Specification references can be checked in accordance with the provisions of section
7342 8.4.4 and its subsections.

7343

7344 Finally, a CPA has several elements whose values are not typically derived from either CPPs
7345 (and can need checking when using a CPA template as the basis for a draft CPA.) The **Status**,
7346 **Start**, **End**, and possibly a **ConversationConstraints** element need to be added. The attributes,

7347

7348 ***CollaborationProtocolAgreement/@cpaid***,

7349 ***CollaborationProtocolAgreement/@version***,

7350 ***CollaborationProtocolAgreement/Status@value***,

7351 ***CollaborationProtocolAgreement/ConversationConstrain@invocationLimit***, and

7352 ***CollaborationProtocolAgreement/ConversationConstraint@concurrentConversations***,

7353

7354 can also be supplied values as needed.

7355 **Appendix F Correspondence Between CPA and ebXML**
 7356 **Messaging Parameters (Normative)**

7357
 7358 The following table shows the correspondence between elements used in the ebXML Messaging
 7359 Service message header and their counterparts in the CPA.
 7360

Message Header Element / Attribute	Corresponding CPA Element / Attribute
<i>PartyId</i> element	<i>PartyId</i> element; if multiple <i>PartyID</i> elements occur under the same <i>PartyInfo</i> element in the <i>CPA</i> , all of them MUST be included in the <i>Message Header</i>
<i>Role</i> element	<i>Role</i> element
<i>CPAId</i> element	<i>cpaid</i> attribute in <i>CollaborationProtocolAgreement</i> element
<i>ConversationId</i> element	No equivalent; SHOULD be generated by software above the Message Service Interface (MSI)
<i>Service</i> element	<i>Service</i> element
<i>Action</i> element	<i>action</i> attribute in <i>ThisPartyActionBinding</i> element
<i>TimeToLive</i> element	Computed as the sum of <i>Timestamp</i> (in message header) + <i>PersistDuration</i> (under <i>DocExchange/ebXMLReceiverBinding</i>)
<i>MessageId</i> element	No equivalent; generated by the MSH per message
<i>Timestamp</i> element	No equivalent; generated by the MSH per message
<i>RefToMessageId</i> element	No equivalent; usually passed in by the application where applicable; SHOULD be used for correlating response messages with request messages
<i>SyncReply</i> element	<i>syncReplyMode</i> attribute in <i>MessagingCharacteristics</i> element; the <i>SyncReply</i> element is included if and only if the <i>syncReplyMode</i> attribute is not “none”
<i>DuplicateElimination</i> element	<i>duplicateElimination</i> attribute in <i>MessagingCharacteristics</i> element; the <i>DuplicateElimination</i> element is included if the <i>duplicateElimination</i> attribute under <i>MessagingCharacteristics</i> is set to “always”, or if it is set to “perMessage” and the application indicates to the MSH that duplicate elimination is desired

<i>Manifest</i> element	<i>Packaging</i> element; each <i>Reference</i> element under <i>Manifest</i> SHOULD correspond to a <i>SimplePart</i> that is referenced from one of the <i>CompositeList</i> elements under <i>Packaging</i>
<i>xlink:role</i> attribute in <i>Reference</i> element	<i>xlink:role</i> attribute in <i>SimplePart</i> element
<i>AckRequested</i> element	<i>ackRequested</i> attribute in <i>MessagingCharacteristics</i> element; an <i>AckRequested</i> element is included in the SOAP Header if the <i>ackRequested</i> attribute is set to “always”; if it is set to “perMessage”, input passed to the MSI is to be used to determine if an <i>AckRequested</i> element needs to be included; likewise, the signed attribute under <i>AckRequested</i> will be appropriately set based on the <i>ackSignatureRequested</i> attribute and possibly determined by input passed to the MSI
<i>MessageOrder</i> element	<i>messageOrderSemantics</i> attribute in <i>ReliableMessaging</i> element; the <i>MessageOrder</i> element will be present if the <i>AckRequested</i> element is present, and if the <i>messageOrderSemantics</i> attribute in the <i>ReliableMessaging</i> element is set to "Guaranteed"
<i>ds:Signature</i> element	<i>ds:Signature</i> will be present in the SOAP Header if the <i>isNonRepudiationRequired</i> attribute in the <i>BusinessTransactionCharacteristics</i> element is set to “true”; the relevant parameters for constructing the signature can be obtained from the <i>SenderNonRepudiation</i> and <i>ReceiverNonRepudiation</i> elements

7361
7362
7363
7364
7365

The following table shows the implicit parameters employed by the ebXML Messaging Service that are not included in the message header and how those parameters can be obtained from the CPA.

Implicit Messaging Parameters	Corresponding CPA Element / Attribute
<i>Retries</i> (not in Message Header) but used to govern Reliable Messaging behavior in sender	<i>Retries</i> element (under <i>ReliableMessaging</i> element)
<i>RetryInterval</i> (not in Message Header) but used to govern Reliable Messaging behavior in sender	<i>RetryInterval</i> element (under <i>ReliableMessaging</i> element)
<i>PersistDuration</i> (not in Message Header) but used to govern Reliable Messaging behavior in receiver	<i>PersistDuration</i> element (under <i>ebXMLReceiverBinding</i> element)

<i>Endpoint</i> (not in Message Header) but used for sending SOAP message	<i>Endpoint</i> element (under <i>TransportReceiver</i>); the type of message being sent MUST be passed in to the MSI; an appropriate endpoint can then be selected from among the <i>Endpoints</i> included under the <i>TransportReceiver</i> element
Use <i>Service & Action</i> to determine the corresponding <i>DeliveryChannel</i>	<i>DeliveryChannel</i>
Use <i>ReceiverDigitalEnvelope</i> to determine the encryption algorithm and key	<i>ReceiverDigitalEnvelope</i>
Use <i>SenderNonRepudiation</i> to determine signing certificate(s) and <i>ReceiverNonRepudiation</i> to determine the trust anchors and security policy to apply to the signing certificate	<i>SenderNonRepudiation</i> and <i>ReceiverNonRepudiation</i>
Use <i>Packaging</i> to determine how payload containers ought to be encapsulated. Also use <i>Packaging</i> to determine how an individual SimplePart ought to be extracted and validated against its schema	<i>Packaging</i>
Use <i>TransportClientSecurity</i> and <i>TransportServerSecurity</i> to determine certificates to be used by server and client for authentication purposes	<i>TransportClientSecurity</i> and <i>TransportServerSecurity</i>
Use the <i>DeliveryChannel</i> identified by <i>defaultMshChannelId</i> for standalone MSH level messages like Acknowledgment, Error, StatusRequest, StatusResponse, Ping, Pong, unless overridden by <i>OverrideMshActionBinding</i>	<i>defaultMshChannelId</i> attribute in <i>PartyInfo</i> element, and <i>OverrideMshActionBinding</i>

7366 **Appendix G Glossary of Terms**

7367

Term	Definition
AGREEMENT	An arrangement between two partners that specifies in advance the conditions under which they will trade (terms of shipment, terms of payment, collaboration protocols, etc.) An agreement does not imply specific economic commitments.
APPLICATION	Software above the level of the MSH that implements a Service by processing one or more of the Messages in the Document Exchanges associated with the Service.
AUTHORIZATION	A right or a permission that is granted to a system entity to access a system resource.
BUSINESS ACTIVITY	A business activity is used to represent the state of the business process of one of the partners. For instance the requester is either in the state of sending the request, in the state of waiting for the response, or in the state of receiving.
BUSINESS COLLABORATION	An activity conducted between two or more parties for the purpose of achieving a specified outcome.
BUSINESS DOCUMENT	The set of information components that are interchanged as part of a business activity.
BUSINESS PARTNER	An entity that engages in business transactions with another business partner(s).
BUSINESS PROCESS	The means by which one or more activities are accomplished in operating business practices.
BUSINESS PROCESS SPECIFICATION SCHEMA	Defines the necessary set of elements to specify run-time aspects and configuration parameters to drive the partners' systems used in the collaboration. The goal of the BP Specification Schema is to provide the bridge between the eBusiness process modeling and specification of eBusiness software components.
BUSINESS TRANSACTION	A business transaction is a logical unit of business conducted by two or more parties that generates a computable success or failure state. The community, the partners, and the process, are all in a definable, and self-reliant state prior to the business transaction, and in a new definable, and self-reliant state after the business transaction. In other words if you are still 'waiting' for your business partner's response or reaction, the business transaction has not completed.
CLIENT	Software that initiates a connection with a <i>Server</i> .
COLLABORATION	Two or more parties working together under a defined set of rules.

COLLABORATION PROTOCOL	The protocol that defines for a Collaborative Process: 1. The sequence, dependencies and semantics of the Documents that are exchanged between Parties in order to carry out that Collaborative Process, and 2. The Messaging Capabilities used when sending documents between those Parties. Note that a Collaborative Process can have more than one Collaboration Protocol by which it can be implemented.
COLLABORATION PROTOCOL AGREEMENT (CPA)	Information agreed between two (or more) Parties that identifies or describes the specific Collaboration Protocol that they have agreed to use. A CPA indicates what the involved Parties “will” do when carrying out a Collaborative Process. A CPA is representable by a Document.
COLLABORATION PROTOCOL PROFILE (CPP)	Information about a Party that can be used to describe one or more Collaborative Processes and associated Collaborative Protocols that the Party supports. A CPP indicates what a Party “can” do in order to carry out a Collaborative Process. A CPP is representable by a Document. While logically, a CPP is a single document, in practice, the CPP might be a set of linked documents that express various aspects of the capabilities. A CPP is not an agreement. It represents the capabilities of a Party.
COLLABORATIVE PROCESS	A shared process by which two Parties work together in order to carry out a process. The Collaborative Process can be defined by an ebXML Collaboration Model.
CONFORMANCE	Fulfillment of a product, process or service of all requirements specified; adherence of an implementation to the requirements of one or more specific standards or technical specifications.
DIGITAL SIGNATURE	A digital code that can be attached to an electronically transmitted message that uniquely identifies the sender
DOCUMENT	A Document is any data that can be represented in a digital form.
DOCUMENT EXCHANGE	An exchange of documents between two parties.
ENCRYPTION	Cryptographic transformation of data (called "plaintext") into a form (called "ciphertext") that conceals the data's original meaning to prevent it from being known or used. If the transformation is reversible, the corresponding reversal process is called "decryption", which is a transformation that restores encrypted data to its original state.
EXTENSIBLE MARKUP LANGUAGE	XML is designed to enable the exchange of information (data) between different applications and data sources on the World Wide Web and has been standardized by the W3C.
IMPLEMENTATION	An implementation is the realization of a specification. It can be a software product, system or program.
MESSAGE	The movement of a document from one party to another.

MESSAGE HEADER	A specification of the structure and composition of the information necessary for an ebXML Messaging Service to successfully generate or process an ebXML compliant message.
MESSAGING CAPABILITIES	The set of capabilities that support exchange of Documents between Parties. Examples are the communication protocol and its parameters, security definitions, and general properties of sending and receiving messages.
MESSAGING SERVICE	A framework that enables interoperable, secure and reliable exchange of Messages between Trading Partners.
PACKAGE	A general-purpose mechanism for organizing elements into groups. Packages can be nested within other packages.
PARTY	A Party is an entity such as a company, department, organization or individual that can generate, send, receive or relay Documents.
PARTY DISCOVERY PROCESS	A Collaborative Process by which one Party can discover CPP information about other Parties.
PAYLOAD	A section of data/information that is not part of the ebXML wrapping.
PAYLOAD CONTAINER	A container used to envelope the real payload of an ebXML message. If a payload is present, the payload container consists of a MIME header portion (the ebXML Payload Envelope) and a content portion (the payload itself).
PAYLOAD ENVELOPE	The specific MIME headers that are associated with a MIME part.
RECEIVER	Recipient of a <i>Message</i> .
REGISTRY	A mechanism whereby relevant repository items and metadata about them can be registered such that a pointer to their location, and all their metadata, can be retrieved as a result of a query.
REQUESTER	Initiator of a <i>Business Transaction</i> .
RESPONDER	A counterpart to the initiator in a <i>Business Transaction</i> .
ROLE	The named specific behavior of an entity participating in a particular context. A role could be static (e.g., an association end) or dynamic (e.g., a collaboration role).
SECURITY POLICY	A set of rules and practices that specify or regulate how a system or organization provides security services to protect sensitive and critical system resources.
SENDER	Originator of a <i>Message</i> .
SERVER	Software that accepts a connection initiated by a <i>Client</i> .

UNIQUE IDENTIFIER	The abstract concept of utilizing a standard mechanism and process for assigning a sequence of alphanumeric codes to ebXML Registry items, including: Core Components, Aggregate Information Entities, and Business Processes.
UNIVERSALLY UNIQUE IDENTIFIER (UUID)	An identifier that is unique across both space and time, with respect to the space of all UUIDs. A UUID can be used for multiple purposes, from tagging objects with an extremely short lifetime, to reliably identifying very persistent objects across a network.

7368

7369